UNIVERSITY OF PIRAEUS

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGIES

DEPARTMENT OF INFORMATICS

PhD Thesis

Thesis title:	(English) Distributed security and trust management in multi-authority and multi-domain environments based on blockchain. Case studies in healthcare and supply chain management systems (Greek) Κατανεμημένη ασφάλεια και διαχείριση εμπιστοσύνης σε περιβάλλοντα πολλαπλών αρχών και πολλαπλών τομέων βασισμένα στο Blockchain. Μελέτες περίπτωσης σε συστήματα διαχείρισης δεδομένων υγείας και εφοδιαστικής αλυσίδας
Student's name-surname:	Evangelos Malamas
Father's Name	Nikolaos Malamas
Student's ID No:	ΠΛΔ 1804
Main Supervisor:	Kotzanikolaou Panayiotis, Professor

June 2024

PhD Thesis
was prepared during the Programme of Doctoral Studies
of the Department of Informatics of the School of Information and Communication Technologies of the University of Piraeus
for the degree of Doctor of Philosophy

3-Member Supervising Committee

Panayiotis Kotzanikolaou
Professor
University of Piraeus
School of Information and Communication Technologies
Department of Informatics
(Supervisor)

Mike Burmester
Professor
Florida State University
Computer Science Department
(Member)

Sokratis Katsikas
Professor
Norwegian University of Science and Technology
Department of Information Security and Communication Technology
(Member)

PhD Thesis
was presented before the 7-Member Examination Committee and approved on June 20, 2024

7-Member Examination Committee

Panayiotis Kotzanikolaou
Professor
University of Piraeus
School of Information and Communication Technologies
Department of Informatics

Mike Burmester
Professor
Florida State University
Computer Science Department

Sokratis Katsikas
Professor
Norwegian University of Science and Technology
Department of Information Security and Communication Technology

Christos Douligeris
Professor
University of Piraeus
School of Information and Communication Technologies
Department of Informatics

Constantinos Patsakis
Associate Professor
University of Piraeus
School of Information and Communication Technologies
Department of Informatics

Mihalis Psarakis
Associate Professor
University of Piraeus
School of Information and Communication Technologies
Department of Informatics

Thomas Dasaklis
Assistant Professor
Hellenic Open University
School of Social Sciences

ACKNOWLEDGMENTS

Completing this PhD has been a significant milestone, and it would not have been possible without the support and guidance of many individuals.

First and foremost, I extend my deepest gratitude to my advisor, Prof. Panayiotis Kotzanikolaou, whose insightful guidance and unwavering patience have been invaluable throughout my research journey. Your expertise and encouragement have been instrumental in shaping this thesis. I am also profoundly thankful to my co-supervisors, Prof. Mike Burmester and Prof. Sokratis Katsikas, for their support and valuable insights. Your mentorship has greatly enriched my academic experience. Special thanks to the members of my Evaluation Committee, Prof. Christos Douligeris, Associate Prof. Constantinos Patsakis, Associate Prof. Mihalis Psarakis, and Assistant Prof. Thomas Dasaklis. I am especially grateful to my colleagues and collaborators, Assistant Prof. Thomas Dasaklis, Dimitris Koutras and Filippos Fotopoulos, for their contributions and for sharing this research journey with me. Your dedication and teamwork have been a source of inspiration.

Lastly, I am forever indebted to my family and friends. Their support, love, and encouragement have been my foundation and motivation through the challenging times of this PhD journey. Thank you for believing in me and for always being there.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Today, our digital lives are increasingly governed by decentralized systems that manage massive amounts of sensitive and non-sensitive data. The vast diffusion of IoT is shaping networks of multiple interconnected devices that provide services and collect data by interacting with each other. This technology has paved the way for increased efficiency, automation and data-driven decision-making in manufacturing, healthcare, and supply chain, among other industries. Multi-authority and multi-domain ecosystems are becoming increasingly common in these decentralized systems, as different organizations and domains collaborate to provide more comprehensive and efficient services.

The integration of multiple authorities and domains in decentralized systems presents significant security and trust challenges. With data and services distributed across several nodes and domains, these systems are vulnerable to security and trust threats, including unauthorized access, data breaches, identity theft, ransomware exploits. Cybercriminals find the management of large amounts of personal and business information lucrative, making these systems a prime target for attacks. Furthermore, the requirement for greater interoperability and fine-grained information sharing among participants further multiplies these risks. Therefore, Trust Management (TM) is essential to ensure trustworthy data fusion and mining, certified services, and improved user privacy and information security in such complex ecosystems.

However, traditional TM infrastructures are unable to resolve issues such as the need for fine-grained access control based on automated enforcement of defined policies due to the cyber-physical and decentralized nature of these systems. As a result, novel TM solutions that take advantage of techniques such as blockchain are necessary to enable automated and trustworthy interactions between participants while incorporating privacy-preserving

mechanisms.

The shift from centralized to decentralized multi-authority and multi-domain environments presents unique challenges that must be addressed. To facilitate secure and reliable communication and information exchange, it is crucial to have a distributed trust management system in place. This system must be capable of managing trust between multiple domains, each with its own set of authorities, policies, and security requirements. As the digital landscape becomes increasingly complex, and the threat of cyber attacks continues to rise, it is imperative to have a robust and secure trust management system that can protect against malicious actors and prevent unauthorized access to sensitive data. Using the principles of distributed systems, a trust management system can provide a higher level of security, scalability, and dependability, making it an indispensable component of modern digital infrastructures.

The main research goal of this Ph.D. thesis is to contribute to the understanding of the security requirements of multi-authority and multi-domain ecosystems and also propose novel security mechanisms that support trust among the participants.

The Thesis has four sections, each of which includes several chapters. In Section I the introduction (Chapter 1) and the review of the relevant literature (Chapter 2) are introduced, in order to present the current state-of-the-art and the open research challenges related to the distributed Trust Management for multi-authority and multi-domain ecosystems. Section II (Chapters 3-4-5-6) presents the novel hierarchical multi-blockchain solution for trust management and access control for multi-authority and multi-domain environments. In particular, in Chapter 3 we describe the Hierarchical Multi-Blockchain Access Control (HMBAC) model, in Chapter 4 we present the system design, in Chapter 5 we dive deeper into the implementation aspects of HMBAC presented in the previous chapter by introducing the *Janus* framework. In Chapter 6, we present the security and performance analysis of *Janus*. In Section III, we describe two additional security features for the Janus framework, which extend the novel solution presented in Section II. Specifically, in Chapter 7 we propose a distributed self-sovereign identity (SSI) infrastructure for device authentication. In Chapter 8, we present an extension to connect the proposed framework with legacy ERP systems. Section IV (Chapter 9), summarizes the results of this thesis that are related to the solutions presented in Sections II and III and their validation, along with open research challenges that require additional future work, respectively.

Σήμερα, η ψηφιακή μας ζωή διέπεται όλο και περισσότερο από αποκεντρωμένα συστήματα που διαχειρίζονται τεράστιες ποσότητες ευαίσθητων και μη ευαίσθητων δεδομένων. Η τεράστια διάδοση του IoT διαμορφώνει δίκτυα πολλαπλών διασυνδεδεμένων συσκευών που παρέχουν υπηρεσίες και συλλέγουν δεδομένα αλληλοεπιδρώντας μεταξύ τους. Αυτή η τεχνολογία έχει ανοίξει το δρόμο για αυξημένη αποδοτικότητα, αυτοματοποίηση και λήψη αποφάσεων βάσει δεδομένων στη μεταποίηση, την υγειονομική περίθαλψη και την εφοδιαστική αλυσίδα, μεταξύ άλλων κλάδων. Τα οικοσυστήματα πολλαπλών αρχών και τομέων γίνονται όλο και πιο συνηθισμένα σε αυτά τα αποκεντρωμένα συστήματα, καθώς διαφορετικοί οργανισμοί και τομείς συνεργάζονται για την παροχή πιο ολοκληρωμένων και αποτελεσματικών υπηρεσιών. Η ενσωμάτωση πολλαπλών αρχών και τομέων σε αποκεντρωμένα συστήματα παρουσιάζει σημαντικές προκλήσεις ασφάλειας και εμπιστοσύνης. Με δεδομένα και υπηρεσίες κατανεμημένα σε πολλαπλούς κόμβους και τομείς, τα συστήματα αυτά είναι ευάλωτα σε διάφορες απειλές ασφάλειας και εμπιστοσύνης, όπως μη εξουσιοδοτημένη πρόσβαση, παραβίαση δεδομένων και κλοπή ταυτότητας. Οι εγκληματίες του κυβερνοχώρου βρίσκουν ελκυστική τη διαχείριση μεγάλου όγκου προσωπικών και επιχειρηματικών πληροφοριών, καθιστώντας τα συστήματα αυτά πρωταρχικό στόχο για επιθέσεις. Επιπλέον, η απαίτηση για μεγαλύτερη διαλειτουργικότητα και λεπτομερή ανταλλαγή πληροφοριών μεταξύ των συμμετεχόντων πολλαπλασιάζει περαιτέρω αυτούς τους κινδύνους. Ως εκ τούτου, η διαχείριση εμπιστοσύνης (ΔΕ) είναι απαραίτητη για την εξασφάλιση αξιόπιστης σύντηξης και εξόρυξης δεδομένων, πιστοποιημένων υπηρεσιών και βελτιωμένης ιδιωτικότητας και ασφάλειας πληροφοριών των χρηστών σε τέτοια πολύπλοκα οικοσυστήματα. Ωστόσο, οι παραδοσιακές υποδομές διαχείρισης εμπιστοσύνης δεν είναι σε θέση να επιλύσουν ζητήματα όπως η ανάγκη για λεπτομερή έλεγχο πρόσβασης με βάση την αυτοματοποιημένη επιβολή καθορισμένων πολιτικών λόγω της κυβερνο-φυσικής και αποκεντρωμένης φύσης αυτών των συστημάτων. Ως αποτέλεσμα, απαιτούνται νέες λύσεις διαχείρισης εμπιστοσύνης που θα αξιοποιούν τεχνικές όπως η αλυσίδα μπλοκ (blockchain) για να επιτρέψουν αυτοματοποιημένες και αξιόπιστες αλληλεπιδράσεις μεταξύ των συμμετεχόντων, ενσωματώνοντας παράλληλα μηχανισμούς διατήρησης της ιδιωτικότητας. Η μετάβαση από κεντρικά σε αποκεντρωμένα περιβάλλοντα πολλαπλών αρχών και πολλαπλών τομέων παρουσιάζει μοναδικές προκλήσεις που πρέπει να αντιμετωπιστούν. Για τη διευκόλυνση της ασφαλούς και αξιόπιστης επικοινωνίας και ανταλλαγής πληροφοριών, είναι ζωτικής σημασίας η ύπαρξη ενός κατανεμημένου συστήματος διαχείρισης εμπιστοσύνης. Το σύστημα αυτό πρέπει να είναι ικανό να διαχειρίζεται την εμπιστοσύνη μεταξύ πολλαπλών τομέων, ο καθένας με το δικό του σύνολο αρχών, πολιτικών και απαιτήσεων ασφαλείας. Καθώς το ψηφιακό τοπίο γίνεται ολοένα και πιο πολύπλοκο και η απειλή των επιθέσεων στον κυβερνοχώρο συνεχίζει να αυξάνεται, είναι επιτακτική ανάγκη να υπάρχει ένα ισχυρό και ασφαλές σύστημα διαχείρισης εμπιστοσύνης που να μπορεί να προστατεύει από κακόβουλους φορείς και να αποτρέπει τη μη εξουσιοδοτημένη πρόσβαση σε ευαίσθητα δεδομένα. Χρησιμοποιώντας τις αρχές των κατανεμημένων συστημάτων, ένα σύστημα διαχείρισης εμπιστοσύνης μπορεί να παρέχει υψηλότερο επίπεδο ασφάλειας, επεκτασιμότητας και αξιοπιστίας, καθιστώντας το απαραίτητο συστατικό των σύγχρονων ψηφιακών υποδομών. Ο κύριος ερευνητικός στόχος αυτής της διδακτορικής διατριβής είναι να συμβάλει στην κατανόηση των απαιτήσεων ασφάλειας σε οικοσυστήματα πολλαπλών αρχών και πολλαπλών τομέων και επίσης να προτείνει νέους μηχανισμούς ασφάλειας που υποστηρίζουν την εμπιστοσύνη μεταξύ των συμμετεχόντων. Η διατριβή διαρθρώνεται σε τέσσερις ενότητες, καθεμία από τις οποίες περιλαμβάνει αρκετά κεφάλαια. Στην Ενότητα Ι παρουσιάζεται η εισαγωγή (Κεφάλαιο 1) και η ανασκόπηση της σχετικής βιβλιογραφίας (Κεφάλαιο 2), προκειμένου να παρουσιαστούν η τρέχουσα κατάσταση και οι ανοικτές ερευνητικές προκλήσεις που σχετίζονται με την κατανεμημένη διαχείριση εμπιστοσύνης για οικοσυστήματα πολλαπλών αρχών και περιοχών. Στην Ενότητα ΙΙ (Κεφάλαια 3-4-5-6) παρουσιάζεται η νέα ιεραρχική λύση πολλαπλών αλυσίδων μπλοκ για τη διαχείριση εμπιστοσύνης και τον έλεγχο πρόσβασης για περιβάλλοντα πολλαπλών αρχών και πολλαπλών τομέων. Ειδικότερα, στο Κεφάλαιο 3 περιγράφουμε το μοντέλο Hierarchical Multi-Blockchain Access Control (HMBAC), στο Κεφάλαιο 4 παρουσιάζουμε τον σχεδιασμό του συστήματος,

στο Κεφάλαιο 5 εμβαθύνουμε στις πτυχές υλοποίησης του HMBAC που παρουσιάστηκαν στο προηγούμενο κεφάλαιο, παρουσιάζοντας το πλαίσιο Janus. Στο Κεφάλαιο 6, παρουσιάζουμε την ανάλυση ασφάλειας και επιδόσεων του Janus. Στο Κεφάλαιο III, περιγράφουμε δύο πρόσθετα χαρακτηριστικά ασφαλείας για το πλαίσιο Janus, τα οποία επεκτείνουν τη νέα λύση που παρουσιάστηκε στο Κεφάλαιο II. Συγκεκριμένα, στο Κεφάλαιο 7 προτείνουμε μια κατανεμημένη υποδομή αυτοκυριαρχούμενης ταυτότητας (Self Sovereign Identity - SSI) για τον έλεγχο ταυτότητας συσκευών. Στο Κεφάλαιο 8, παρουσιάζουμε μια επέκταση για τη σύνδεση του προτεινόμενου πλαισίου με παλαιά συστήματα ERP.

SECTION I
Foundations and Related Work

CHAPTER 1

INTRODUCTION

The Internet of Things (IoT) represents a rapidly evolving frontier, characterized by a vast and complex network of interconnected devices, systems, and networks. This proliferation of IoT devices has revolutionized our ability to collect and analyze data from a multitude of sources in real time. However, this enhanced connectivity also introduces significant security challenges, notably expanding the attack surface and causing numerous trust issues. These challenges are particularly pronounced in environments governed by multiple authorities and domains, such as healthcare, supply chain management, and finance, necessitating robust measures to ensure the security and safe operation of these interconnected systems.

The evolution of the Internet has fundamentally transformed the landscape of resource sharing, shifting from closed, centrally managed, and relatively static local computing environments to open, decentralized, autonomous, and dynamically collaborative inter-domain computing environments. These transformations have introduced a host of challenges, particularly in terms of access to shared resources. Issues such as user authentication management, authorization policy formulation, and other trust management tasks have become increasingly complex, making trust management and security paramount in dynamic, distributed multi-domain environments.

In multi-authority and multi-domain contexts, ensuring trust and security is of utmost importance. Diverse organizations, each with its own objectives, policies, and security protocols, are required to collaborate and share information, often leading to conflicting security and trust requirements. This complexity complicates the protection of confidentiality, privacy, and the integrity of sensitive data. For example, in the healthcare sector, the sharing of medical data between different organizations and domains introduces significant vulnerabilities to privacy and security. In supply chain management, the integration of IoT devices and systems in various domains increases security vulnerabilities and obscures the visibility of the supply chain, challenging the assurance of the authenticity and traceability of the product. Similarly, in the financial sector, the incorporation of IoT technologies into financial transactions raises the risks of fraud and financial crimes.

This Thesis seeks to tackle the intricate challenges of security and trust in multi-authority and multi-domain environments through the exploration of distributed security and trust management solutions. Our research is centered on the design, implementation, and evaluation of a hierarchical multi-blockchain access control model, aimed at facilitating secure and efficient trust management across environments with multiple stakeholders. In addition, we introduce two novel security enhancements to augment the proposed system. The first enhancement involves a distributed self-sovereign identity (SSI) framework for users and devices, which serves as an advanced identity management module. The second, is a comprehensive solution for the integration of the proposed system with existing legacy systems, incorporating a crucial adaptation phase. Overall, the proposed framework is designed to secure device connectivity and thwart attacks by leveraging the hierarchical multi-blockchain architecture, thereby offering a comprehensive solution to the emerging security threats in IoT ecosystems.

1.1 A Paradigm Shift: From Centralized to Distributed Multi-authority and Multi-domain Environments

The evolution of digital systems, marked by the transition from centralized architectures to distributed, multi-authority, and multi-domain environments, represents a paradigm shift that has fundamentally altered how data are managed, shared, and secured. This transformation is characterized by a move away from monolithic, single-entity-controlled systems towards more collaborative, decentralized networks that involve multiple stakeholders.

Historically, the centralized model dominated the digital landscape, offering a straightforward but limited approach to system management. These centralized systems, while easy to manage, suffered from significant drawbacks such as scalability issues, susceptibility to single point of failure, and lack of flexibility, which became increasingly untenable with the growth in data volume and user base. The limitations of centralized systems catalyzed the shift towards more distributed architectures, a change enabled by breakthroughs in distributed computing, blockchain technology, and advancements in cryptographic techniques and consensus algorithms. This shift heralded a new era of digital systems, characterized by a dispersion of control and collaborative operation across diverse entities.

In distributed multi-authority and multi-domain environments, the architecture is fundamentally decentralized, distributing control across multiple nodes or entities that operate under their own authority but work collaboratively. This decentralization mitigates the risks associated with single points of failure and enhances the system's resilience. Each domain within such an environment retains autonomy over its data, policies, and security measures, enabling a customized approach that meets specific needs or regulatory requirements. Despite their diverse and autonomous nature, these systems are designed for interoperability, facilitating seamless data exchange and collaboration between different domains and authorities. This scalability and flexibility allow the systems to easily accommodate growth and adapt to changing conditions without centralized coordination.

The transition to distributed multi-authority and multi-domain systems offers several notable advantages. Enhanced security emerges from the distribution of trust among multiple entities, reducing the vulnerability inherent in centralized systems, such as the single point of failure. The decentralized structure makes it more challenging for attackers to compromise the entire network, offering a robust defense against cyber threats. Interoperability is another significant benefit, as it enables entities across different domains to share information and collaborate more effectively, driving innovation and improving service delivery. The resilience of these systems is markedly improved, with their ability to maintain continuous operation and reliability in the face of attacks, failures, or disruptions. Furthermore, the inherent flexibility of distributed architectures allows for more agile responses to changes in regulatory environments, technological advancements, and evolving user requirements, providing a dynamic framework for future digital ecosystems.

This paradigm shift towards distributed, multiauthority, and multidomain environments not only addresses the shortcomings of centralized models, but also paves the way for a future characterized by more open, collaborative, and secure digital interactions. As these environments continue to develop, they promise to fundamentally alter the landscape of digital systems, offering a foundation for innovation, efficiency, and scalability that will shape the digital age.

The transition to distributed multi-authority and multi-domain systems, while offering numerous advantages over traditional centralized models, introduces a unique set of challenges, particularly in the realms of security and trust. These systems, by their very nature, involve a complex web of interactions among various entities, each with their own objectives, policies, and security requirements. This complexity necessitates novel approaches to enforce trust and ensure secure access to shared data among participants who may not inherently trust each other.

One of the main challenges in these environments is ensuring the security of the interactions and transactions that take place in this distributed landscape. The decentralized nature of these systems, although beneficial for resilience and reducing single points of failure, complicates the enforcement of uniform security policies and practices. Each entity in the network may employ different security protocols, leading to potential vulnerabilities at the interfaces where these diverse systems interact. Furthermore, the open and distributed architecture makes it challenging to detect and mitigate security threats, such as unauthorized access or data breaches, which can proliferate rapidly across the network due to its interconnected nature.

Trust management is another significant challenge in distributed multi-authority and multi-domain systems. In an environment where participants are autonomous and may not have pre-existing trust relationships, establishing a framework for mutual trust becomes critical. Traditional centralized trust models, which rely on a single trusted authority, are not applicable in this decentralized context. Instead, there is a need for mechanisms that can dynamically establish and manage trust among a multitude of untrusted parties, taking into account the varying degrees of trustworthiness and the changing dynamics of the network.

Addressing these challenges requires the development of innovative mechanisms that can both enforce trust and provide secure, fine-grained access control over shared data. These mechanisms must be capable of operating in a decentralized environment, where they can facilitate secure interactions without relying on a central authority. A promising approach is the use of distributed ledger technologies, such as blockchain, which can provide a transparent and immutable record of transactions, thus underpinning trust among participants. However, blockchain alone may not address all security and trust issues, particularly in terms of fine-grained access control and dynamic management of trust relationships.

To overcome these limitations, there is growing interest in advanced cryptographic techniques, such as attribute-based encryption (ABE), which can offer more nuanced access control mechanisms. ABE enables data encryption in such a way that only users with specific attributes or credentials can decrypt and access the information, allowing fine-grained control over who can see what data. Additionally, the development of decentralized identity and reputation systems can further enhance trust management by providing verifiable and portable identities for participants, along with mechanisms to assess and establish the trustworthiness of entities based on their behavior and interactions within the system.

As a result of the preceding discussion, more emphasis is required on innovative, decentralized, interoperable, and adaptable trust management systems. The following are two motivational examples.

Example 1: Fine-grained access to healthcare data. The medical sector exemplifies a complex, multi-authority and multi-domain ecosystem, encompassing a diverse array of domains including regulatory bodies, healthcare facilities, manufacturing entities, and insurance providers. This sector's structure is inherently intricate due to the interdependencies and interactions among these varied stakeholders, each of whom plays a critical role in the healthcare delivery matrix. At the same time, privacy-sensitive health-related data may be created by various medical IoT devices, such as health monitoring devices (e.g., glucose level, blood pressure, or sleep monitoring systems) or treatment devices (e.g., medical infusion pumps). Users of one authority may require granular access to data maintained by multiple authorities (stakeholders) and domains. For example, while a doctor is on duty, they may require access to the full medical history data maintained in multiple hospital databases, for a patient under emergency treatment, or an administrator of a medical device manufacturer may require access to the configuration data of connected medical devices installed in different hospitals. In addition, regulators may require that the data be accessible from a single entrance platform, to log all data access requests, and monitor privacy violations. At the same time, new sectors or stakeholders can dynamically join or leave the system. Note that users who are simultaneously members of multiple authorities may require special access. For example, a doctor in a hospital may also be a researcher at a university. This doctor would also require access to (statistical) health data maintained in *all* hospitals, for research purposes.

Example 2: Fine-grained access to data in a multi-domain supply chain. Another typical example is the supply chain environment. Consider a distributed supply chain tracking system, collectively used by supply chain stakeholders for collecting, integrating, and analyzing data from a variety of sources. The various stakeholders have different requirements for data access. For example, a container shipping company requires access to data on cargo weight and quantity, while a retail end-user requires access to data on product provenance, storage, and transportation conditions (especially for sensitive merchandise). These systems must be interoperable, but also provide granular access to data. In addition, authorities such as customs or other governmental agencies may also require a single point of access for all queries to data, for global access verifiability, i.e., it must be possible for any entity, either inside or outside the system, to verify all access attempts to the data (either successful or not).

1.2 Trust and Security Requirements in Multi-Authority and Multi-Domain Environments: Research Gaps and Thesis Contribution

The proliferation of distributed multi-authority and multi-domain environments necessitates a reevaluation of traditional trust and security paradigms. As these environments become increasingly prevalent, they reveal significant research gaps that must be addressed to ensure their secure and trustworthy operation and also cover the synchronous needs. This section outlines key areas where existing research falls short, linking to relevant chapters that delve deeper into proposed solutions.

Research Gap 1: Secure Interoperability of Trust Infrastructures in Multi-Authority and Multi-Domain Environments *"A primary concern in environments with multiple authorities and domains is the lack of secure interoperability among diverse trust infrastructures and untrusted parties. These environments are characterized by the coexistence of multiple trust models, which can lead to inconsistencies and vulnerabilities at the intersections of these systems."*

Our Contribution: To address this gap, we propose a system that uses multiple blockchains that leverage cryptographic mechanisms to ensure seamless and secure interoperability among different trust infrastructures. The system allows all stakeholders to manage trust within their domain in a distributed and self-sustaining fashion. For example, it is possible for each stakeholder to internally manage the issuing, updating, or revoking of access credentials. At the same time, the system allows credential interoperability, i.e. credentials issued from independent authorities are mutually trusted, without assuming a globally trusted root authority. For a detailed exploration of this solution, refer to Chapters

3 and 4, where we discuss the architectural and protocol design.

Research Gap 2: Self-managed Identities and Credentials. *"Traditional identity management systems, often centralized, pose significant risks in Multi-Authority and Multi-Domain environments, including privacy concerns and single points of failure. There is a clear need for mechanisms that allow individuals and devices to manage their identities and credentials autonomously, enhancing privacy and control."*

Our Contribution: Our research introduces a model for self-managed identities based on decentralized identity technologies, such as blockchain and decentralized identifiers (DIDs). This model supports the secure and autonomous management of digital identities, enabling entities to control their credentials without relying on a central authority. Chapter 7 delves into the technical underpinnings of this model, including the cryptographic techniques that ensure privacy, and the protocols for identity verification and credential exchange.

Research Gap 3: Fine-Grained Access, Privacy-Preserving Encryption, and Immutable Logging *"Ensuring fine-grained access control, while preserving privacy and maintaining an immutable record of transactions, presents a complex challenge in Multi-Authority and Multi-Domain environments. Existing solutions often fail to adequately balance these requirements, leading to either overly restrictive access controls or insufficient privacy guarantees."*

Our Contribution: To bridge this gap, we propose an integrated solution that combines attribute-based encryption (ABE) and blockchain technology to ensure that all entities/stakeholders have granular access to data and services at a domain level (inter and cross-domain) and at a role level. In addition, temporal access is possible for cases that may need to allow temporary full access to specific data for a specific type of user (e.g. a doctor working in the emergency department). This temporal access is easily revocable. Finally, the system also includes access controls that bypasses avoidance mechanisms and immutable logging of access and transactions. This approach ensures that only authorized entities can access specific data, while also safeguards the privacy of sensitive information and creates a tamper-proof audit trail. Chapters 5 and 6 provides an in-depth analysis of this integrated solution, including its implementation and evaluation in real-world scenarios.

Research Gap 4: Interconnection with legacy system.: *In many ecosystems characterized by the presence of multiple authorities and domains, established systems to automate services are already in place. The introduction of new, decentralized systems—despite their considerable advantages—tends to be met with skepticism by businesses and organizations. This reluctance is primarily attributed to concerns regarding the potential financial outlays and the prospect of entirely replacing existing infrastructures. Current solutions frequently prove inadequate in addressing these apprehensions, posing a significant barrier to the widespread adoption of innovative technologies.*

Our Contribution: This research contributes to the field by developing a blockchain-based architecture designed to integrate seamlessly with existing Enterprise Resource Planning (ERP) systems, thereby addressing the significant gap in the adoption of decentralized systems within multi-authority ecosystems. By building on top of established ERP infrastructure, our approach mitigates the primary concerns associated with the introduction of new technologies—namely, the substantial costs and the potential need for a complete infrastructure overhaul. Our work not only extends the capabilities of traditional systems with the benefits of decentralization but also offers a practical pathway for the adoption of blockchain technology, thereby overcoming a crucial barrier to its wider implementation. This strategic integration exemplifies a forward-thinking response to the complexities of modern business ecosystems, showcasing a scalable model that balances innovation with operational continuity. The proposed architecture is presented in Chapter 8.

By addressing these research gaps with innovative solutions, this thesis aims to advance the field of trust and security in multi-authority and multi-domain environments, laying the groundwork for more secure, efficient, and user-centric digital ecosystems.

1.3 Thesis Structure

In the following paragraphs, we describe from a high-level the layout of the Thesis. For a better understanding, we have structured our work into ten Chapters that are grouped in four main Sections. In particular:

Section I: Foundations and Related Work

- In **Chapter 1**, we delineate the extensive scope of our research, which investigates complex ecosystems characterized by multiple principles and domains. This section elucidates the comprehensive range of knowledge gaps identified within the field and articulates the specific contributions of the thesis to bridging these gaps. By establishing a clear framework for our study, we set the foundation for a detailed exploration of the interdependencies and dynamics within multifaceted systems, highlighting how our findings advance the existing body of knowledge and propose practical solutions to longstanding challenges.

– In **Chapter 2**, we provide a detailed exposition of the technological underpinnings pertinent to our research. Specifically, we delve into critical aspects of blockchain technology, focusing on the functionalities and implications of Smart Contracts. Furthermore, we examine the two principal blockchain networks employed in our study: Ethereum and Hyperledger. This discussion is complemented by a thorough analysis of the extant literature within key research domains relevant to our investigation, namely fine-grained access control, distributed trust management, and privacy-preserving encryption. These areas are explored in the context of the ecosystems under consideration, ensuring a comprehensive understanding of how these technologies interact and the challenges they address. This foundational knowledge sets the stage for subsequent chapters, where the application of these technologies in complex multi-domain environments is articulated.

Section II: Hierarchical Multi Blockchain

– In **Chapter 3**, we introduce a novel Hierarchical Access Control Model specifically engineered to manage access effectively within environments characterized by multiple authorities and domains. This model capitalizes on the decentralized attributes of blockchain technolo-

Sections	Chapter	Title	Contributing paper(s)	Contribution
I.Foundations and related work	1	Introduction		Describes the landscape of multi-authority and multi-domain systems
	2	Review of the literature	[1, 2, 3]	Presents background information about the blockchain and also survey of existing studies.
II.Hierarchical Multi Blockchain	3	HMBAC model	[4]	Presents the hierarchical multi-blockchain access control model
	4	System design	[1, 5]	Analyzes the various components and the design methodology
	5	Implementation	[4]	Develops the HMBAC system, describes of the technical aspects
	6	Security Analysis and Performance Analysis	[4, 5]	Validates the solution's effectiveness and provides a security analysis.
III.Other security features	7	Self-sovereign identity	[6]	Develops an SSI addition to the main system for handling identities
	8	Interconnection with legacy systems	[7, 8, 9]	Presents a blockchain-based solution to interconnect legacy ERP systems
IV.Conclusions and future work	10	Conclusion and future work	[4, 6, 10]	Concludes our work and presents the limitations of the thesis and future work.

Table 1.1: Thesis structure with the corresponding published papers

gies, incorporating a structured hierarchical framework to enhance efficiency, scalability, and security in access control operations.

– In **Chapter 4**, we delineate the architectural design of the proposed hierarchical multi-blockchain framework, which is predicated on the HMBAC model. This chapter details the principal components of the system and outlines the fundamental security features inherent in the architecture.

– In **Chapter 5**, we delve deeper into the implementation details of the proposed framework, describing the system's functionality through an in-depth presentation of the relevant APIs and Smart Contracts. This chapter provides a comprehensive examination of how these components interact to support and enhance the framework's operational capabilities.

– In **Chapter 6**, we present the results of the security and performance analyzes carried out in the proposed framework. Initially, we explore the scalability of the system with respect to its management capabilities. Subsequently, we benchmark the system performance in various configurations and access request rates, providing a thorough evaluation of its operational efficiency.

Section III: Other security features

– In **Chapter 7**, we propose a self-sovereign identity management subsystem, designed to effectively handle device IDs in distributed environments. This subsystem is introduced as a complementary addition to the framework presented in Section II, enhancing its ability to manage identities autonomously.

– In **Chapter 8**, we present a novel architecture that integrates blockchain technology with legacy Enterprise Resource Planning (ERP) systems. The proposed architecture enhances these legacy systems by incorporating a blockchain service layer atop the existing infrastructure. This addition facilitates improved interconnection of shareable data and enhances trust among multiple stakeholders.

Section IV: Conclusions and future work

– In **Chapter 9**, we summarize the key findings of our research and the implications of the proposed hierarchical multi-blockchain access control model. We also outline the limitations encountered during the study. Building upon this groundwork, we propose future research directions to extend and refine the capabilities of our framework, highlighting potential areas for technological advancement and practical application.

CHAPTER 2

LITERATURE REVIEW

In this thesis, we introduce a novel distributed trust management and access control framework tailored for multi-domain and multi-authority settings. This framework leverages the robust security features of blockchain technology, augmented by the deployment of Smart Contracts across a distributed network. Furthermore, it incorporates Multi-Authority Attribute-Based Encryption (MA-ABE) cryptographic schemes to enhance access control. This chapter provides essential background information on the primary technology underpinning the development of the distributed trust management and access control framework, namely blockchain. Additionally, it includes a comprehensive literature review on fine-grained access control and distributed trust management mechanisms, as well as on encryption methods apt for environments characterized by multiple authorities and domains.

2.1 Background

This section explores the fundamental principles underlying blockchain technology. Initially, we elucidate the operational mechanisms of these distributed networks and present their taxonomy based on their access permissions and governance models. Subsequently, we highlight the transformative introduction of Smart Contracts, which automate the enforcement and execution of contractual terms upon the fulfillment of predetermined conditions. Furthermore, we outline the principal characteristics of two well known blockchain networks: Ethereum and Hyperledger.

This foundational knowledge is essential for a thorough comprehension of the chapters that follow, wherein the developed mechanisms are examined in depth. The objective of establishing this context is to augment the reader's understanding of the complex functions and subtleties of the proposed system as detailed in subsequent sections.

2.1.1 Blockchain fundamentals

In 1991, Stuart Haber and W. Scott Stornetta laid the groundwork for blockchain technology with the development of a cryptographically secure chain of blocks, documented in [11]. Their collaboration with Bayer in the subsequent year introduced Merkle trees into the design, enhancing the system's efficiency by allowing the aggregation of multiple documents in a single block [12]. This foundational work was further advanced in 2008 with the emergence of the modern distributed blockchain architecture by an anonymous entity known as Satoshi Nakamoto. Nakamoto's seminal whitepaper, "Bitcoin: A Peer-to-Peer Electronic Cash System" [13], effectively resolved the Byzantine Generals Problem (BGP). This problem is crucial in the field of distributed computing and systems. It encapsulates a situation where participants in a network must agree on a single strategy to avoid failure, but some participants may be unreliable or malicious. This problem is metaphorically likened to a group of generals of the Byzantine army encircling a city, needing to agree on a common battle plan. However, due to the presence of traitors within their ranks, establishing trust and a unanimous decision is complex. The BGP fundamentally addresses the issue of achieving consensus in an environment where communication might be untrustworthy. n the context of blockchain technology, the BGP is highly relevant because blockchains are decentralized networks that rely on consensus mechanisms to validate transactions and add new blocks. Solving the BGP ensures that all nodes in the network agree on the valid state of the blockchain, even in the presence of nodes that may act maliciously or spread false information. Blockchain addresses the BGP through innovative consensus mechanisms, such as Proof of Work (PoW) and Proof of Stake (PoS).

The architectural components of blockchain include (i) nodes, (ii) transactions, (iii) blocks, and (iv) the consensus mechanism. At its core, the blockchain comprises transactions that are digitally signed, representing either agreements or asset transfers. These transactions are rigorously validated by entities known as *nodes*, which are essentially participants in the blockchain network. Among them, the full nodes play a vital role, as they are responsible for verifying adherence to all blockchain rules [14]. These full nodes are crucial in maintaining the integrity of the blockchain; they not only validate transactions but also incorporate them into *blocks*. This decentralized system operates without a central data storage manager, and instead, distributes data uniformly across all network nodes.

Transactions are the fundamental building blocks of a blockchain system. They typically consist of a recipient address, a sender address, and a value. The owner initiates a transfer by digitally signing a hash, which is created by combining the previous transaction with the public key of the receiver. Once created, the transaction is publicly announced to the network. Each node in the network independently maintains its own copy of the blockchain, and the current state of the system is determined by sequentially processing each transaction as it appears in the blockchain. Transactions are grouped together and transmitted to each node in the form of a block. As new transactions are disseminated across the network, each node independently verifies and processes them, with each transaction being time-stamped and recorded in a block. For signing and validating transactions, public key cryptography is used, where all transactions submitted to the blockchain are signed by a user's private key and are verified in the blockchain using the user's public key.

In the blockchain ecosystem, *blocks* serve as fundamental data structures that encapsulate two core elements: a block header and a set of transactions. The block header, functioning as metadata, plays a pivotal role in ascertaining the block's validity. It includes the block's hash, the hash of the preceding block, and other crucial elements like the timestamp and nonce. The second component, the body, houses a collection of transactions, each representing a discrete operation within the blockchain network. These blocks are not only created by miners through a process known as mining — which involves the creation and validation of blocks — but are also replicated across all nodes in the network, ensuring a consistent and decentralized ledger. All blocks on the Blockchain are indexed using a Merkle Tree, which is a lightweight digital fingerprint of all transactions within a block. The structural composition of these blocks, along with the Merkle tree structure, is illustrated in Figure 2.1, showcasing the interplay between the block's constituents and its role in the overarching blockchain framework.

A block can be thought of as a digital ledger page that records a number of transactions. Each block is securely linked to its predecessor, forming a chain of blocks, hence the term "blockchain". The block header is akin to a block's identity card, and contains several crucial information but not the transactions themselves. Each block has a header which ensures the block is in the correct sequence and securely connected to the blockchain. In addition, the block hash which is a unique identifier for each block, is generated from the block header's information through a cryptographic process known as hashing. This hash is a digital fingerprint for the block, ensuring its integrity and immutability, where even if a single bit of the block's data changes, the hash will change entirely, signaling a tampering attempt.

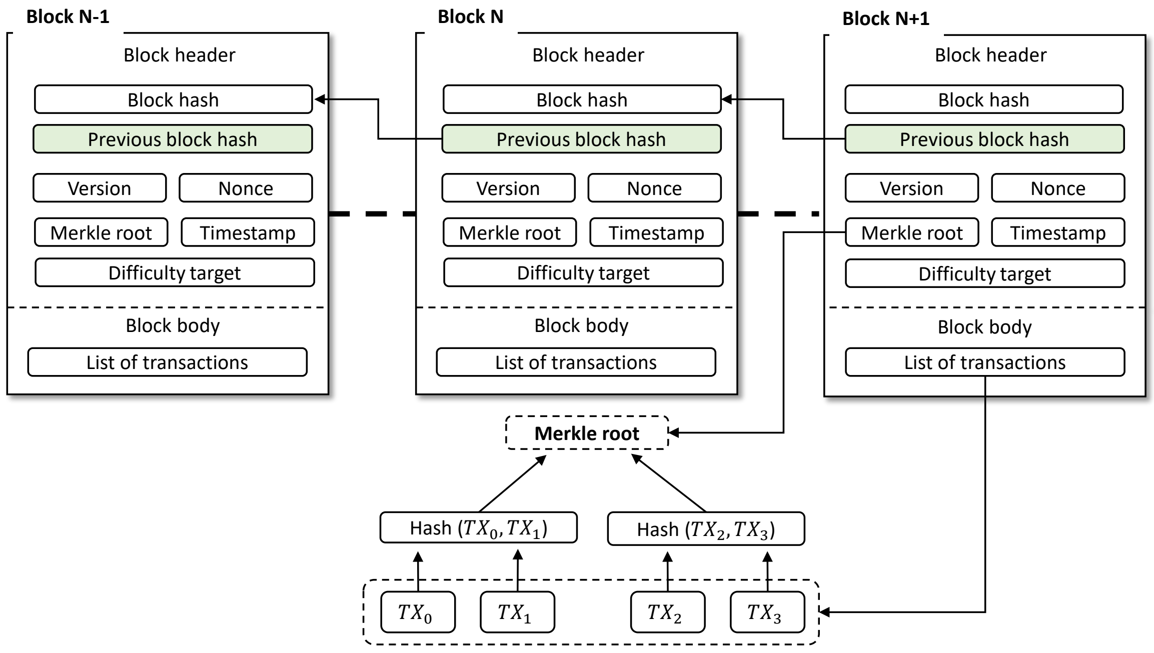


Figure 2.1: Block structure and Merkle trees

In blockchain systems, the *consensus protocols* are mechanisms that ensure unanimity across a decentralized network, validating each transaction and maintaining the integrity of the ledger. For the blockchain to remain secure, it must have a mechanism to prevent a malicious user or group from taking over a majority of validation. These protocols are fundamental in resolving the inherent challenges of distributed computing, particularly in achieving agreement in a trustless environment, and help to overcome challenges like double-spending of digital assets. There are a number of Blockchain consensus mechanisms but regardless of the consensus type used, it is important to note that all transaction data on a chained block is assumed to be trustworthy and the chained data has not been tampered with due to the validation of data by group consensus. Among the various consensus mechanisms, Proof of Work (PoW) and Proof of Stake (PoS) are most prevalent.

Proof of Work (PoW), as employed by Bitcoin, involves a computationally demanding process where miners vie to solve cryptographic puzzles. This system, essentially a cryptographic proof, requires one party (the prover) to demonstrate to others (the verifiers) that substantial computational effort has been expended. This verification is efficiently confirmable by the verifiers with minimal exertion [15]. In PoW, the problem is inherently non-computational, with random guessing being the most effective strategy. Miners attempt to determine the correct "nonce" to validate a block. This involves processing all block data and the nonce through a cryptographic hash function. If the output aligns with the network's current "difficulty" level, the miner succeeds. The network dynamically adjusts this difficulty to maintain equilibrium with the system's overall load. It is important to note that, adding a node to the network increases security by $1/N$ (where N is the number of nodes on the network) and also increases transaction time by $1/N$.

This protocol, while robust in security, faces criticism for its substantial energy consumption and potential environmental impact. It also raises concerns regarding centralization, as entities with significant computational power can dominate the mining process. In contrast, *Proof of Stake (PoS)* offers a more energy-efficient alternative. PoS removes the guessing game from the validation of blocks so mining no longer requires powerful and specialized hardware, therefore, it requires less energy for processing. It selects validators based on their stake in the network, i.e., the amount of cryptocurrency they hold and are willing to 'lock up' as collateral. This method reduces the energy requirement and democratizes the validation process to an extent, as it doesn't require substantial computational power.

Other notable protocols include Delegated Proof of Stake (DPoS), which introduces a democratic voting system to elect validators, and Practical Byzantine Fault Tolerance (PBFT), designed for systems with fewer nodes, focusing on transaction speed and system throughput. Each consensus protocol reflects a trade-off between various factors like security, decentralization, scalability, and energy efficiency. The choice hinges on the specific network's objectives, size, and desired operational efficiency, and also the type of blockchain.

2.1.2 Types of Blockchain

The configuration of a blockchain profoundly influences the nature of the content stored within its blocks, as well as the spectrum of activities undertaken by its diverse participants. Typically, blockchains are architected with distinct objectives in mind, leading to a bespoke design that delineates the types of access and range of tasks allocated to users. This nuanced approach to blockchain design not only tailors the technology to its intended application but also defines the roles and capabilities of its participants, ensuring alignment with the overarching purpose of the blockchain network. In the burgeoning field of blockchain technology, various types of blockchains have emerged, each characterized by distinct features and applications. Predominantly, these types can be categorized into public, private and hybrid blockchains [16]. The basic distinction between the different types is the permission that users have to append and/or read information from the shared ledger. An overview of the architecture for each type is depicted in Fig. 2.2.

When we characterize a blockchain as *public*, we describe fully open public ledgers with the absence of any restrictions on reading and writing permissions for the users. Such permissionless systems permit any individual to connect, access, and contribute to the network, fostering a truly decentralized environment [17]. The intrinsic open nature of these ledgers necessitates the reliance on cryptoeconomic mechanisms within the consensus protocol, a critical process that validates and integrates new blocks while ensuring consistency with the existing chain. Protocols like Bitcoin[1], Ethereum[2], and Monero[3], grounded in Proof of Work (PoW) algorithms, exemplify this model. These platforms allow anyone to download the code, run a public node, and partake in the consensus process. Moreover, public blockchains guarantee the inclusion of valid transactions from any global participant, coupled with the transparency of transactions on public block explorers. Despite this transparency, the system maintains anonymity or pseudonymity for its users, striking a balance between openness and privacy.

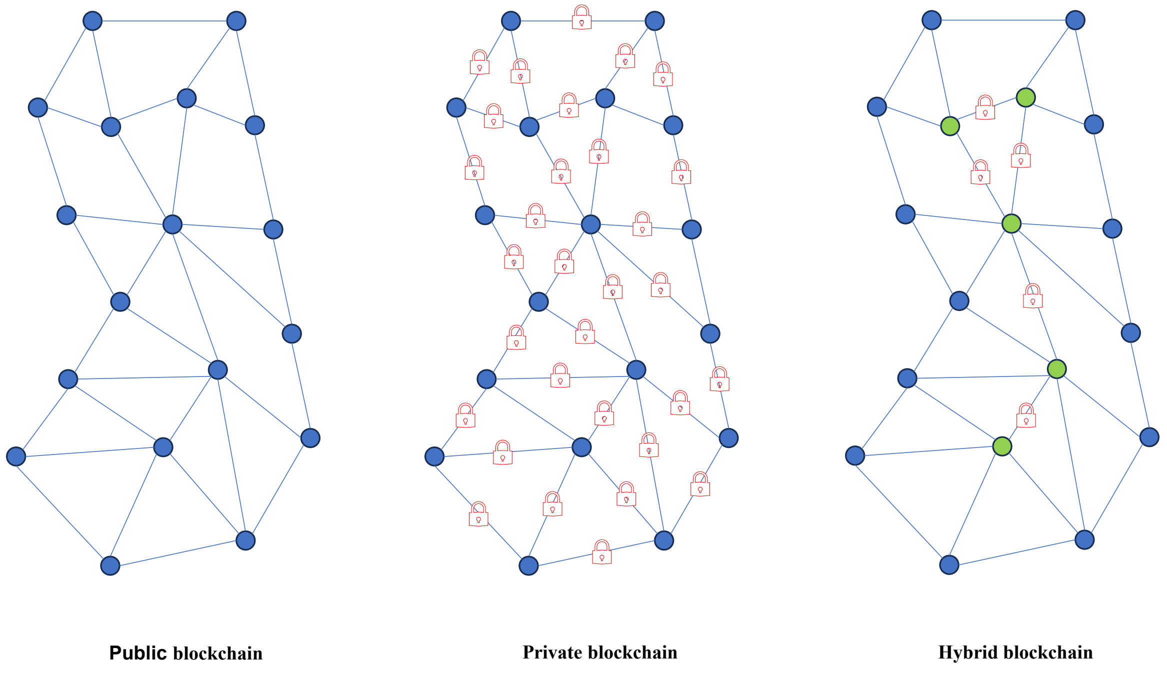


Figure 2.2: Blockchain types high-level architecture

On the other hand, *private* blockchains represent a paradigm markedly distinct from their public counterparts, characterized by restricted reading and writing permissions and a

[1]https://bitcoin.org/en/
[2]https://ethereum.org/en/
[3]https://www.getmonero.org/

more centralized control mechanism. Such permissioned systems, typically limit the modification, addition, or access to information to a select group of participants, often within a single organization [18]. The necessity for a consensus protocol in private blockchains is often obviated by the inherent trust among the nodes. Private ledgers are lauded for their capabilities in facilitating rapid access to information, reducing transaction costs, and offering customizable privacy levels. These systems find pertinent applications in areas such as database management and auditing, predominantly within the confines of a singular entity. The need for public readability is often non-essential in these applications, though in certain instances, public auditability may be desirable. Private blockchain platforms like Monax[4] and Multichain[5] exemplify the utilization of blockchain technology in creating closed ecosystems where transaction verification is internal. While this approach mirrors centralized systems, potentially inheriting similar security vulnerabilities, it offers notable advantages in terms of scalability and compliance with data privacy regulations and other legal mandates. This duality presents a complex landscape where the benefits of blockchain technology are harnessed in a controlled environment, yet it requires careful navigation to mitigate inherent risks.

Another type is *hybrid* blockchains, which may also be called *Federated* or *Consortium*, that incorporate elements of both public and private ledgers. This innovative model delineates a consensus protocol that is typically pre-established and governed by a predetermined consortium of institutions. In such a system, decision-making might, for instance, be distributed among 10 institutions, each controlling a node, with the stipulation that a new block must garner approval from a majority, say 6 institutions, to attain validity. This structure renders the consortium blockchain partially decentralized. The consortium ledger offers a flexible approach to reading permissions, which can be either publicly accessible or confined to a select group of participants. This duality extends to the information itself, where certain data might be made public while other portions remain private. Federated Blockchains, ex-

[4]https://content-blockchain.org/research/monax/
[5]https://multichain.xyz/

emplified by platforms such as R3[6] for banking, EWF[7] for energy and Corda, operate under the auspices of a selected group. These structures restrict participation in the transaction verification process to members of the consortium, excluding general Internet users. This type of Blockchains, characterized by enhanced speed and heightened transaction privacy, predominantly finds its application in the banking sector. The consensus process is meticulously controlled by a pre-selected set of nodes, and the blockchain's readability may either be open to the public or limited to consortium members [19]. This configuration reflects a strategic blend of centralization and decentralization, aiming to leverage the strengths of both public and private blockchain systems. Usually, with the term hybrid we refer to a blockchain controlled by one authority while with the term consortium, we characterize blockchains controlled by a group of authorities.

2.1.3 Smart Contracts

Smart contracts represent a significant innovation in blockchain technology. Smart Contracts were conceived initially by American cryptographer Nick Szabo in 1994. These autonomous computer programs are self-executing codes designed to automatically enforce the terms of a contract, ensuring compliance and execution of agreed-upon conditions. Smart contracts are digitally signed agreements between parties, enforced by virtual software agents, thus removing the need for intermediaries in contract execution [20].

A smart contract's code is deterministic, immutable, and verifiable. Determinism ensures that the contract, when executed on multiple nodes, yields the same outcome for any given input, thereby maintaining consistency and reliability. The immutable nature of the code, once deployed, means it cannot be altered, thus embedding trust but also presenting challenges such as bug fixes and demanding thorough validation. Verifiability allows parties to inspect and confirm the contract code prior to engagement, ensuring transparency.

The advantages of smart contracts over traditional contracts are manifold. The immutability of blockchain technology ensures that contracts cannot be altered once issued,

[6]https://r3.com/

[7]https://www.energyweb.org/

mitigating risks such as financial fraud. By removing intermediaries, smart contracts significantly reduce administrative and service costs. Furthermore, they enhance the efficiency of business processes, exemplified in automated financial settlements in supply chain management upon meeting contractual conditions.

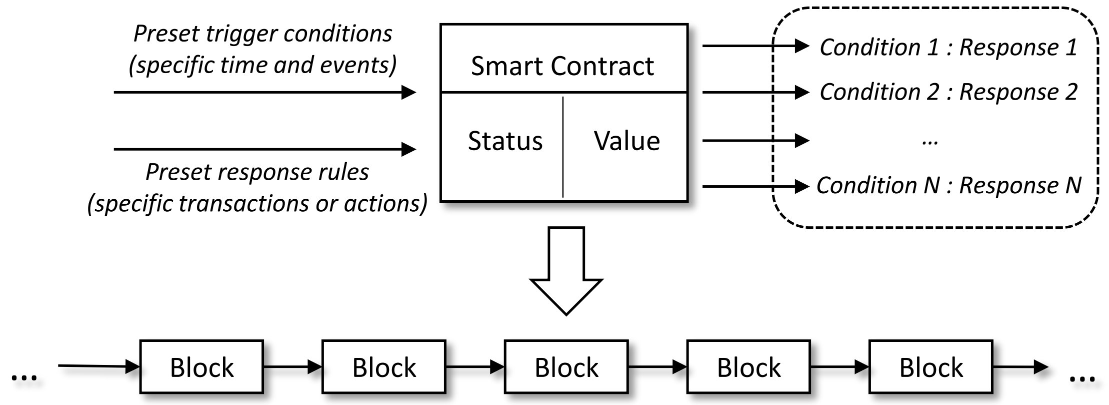


Figure 2.3: Smart Contract operation

The functional mechanism of smart contracts is delineated in Fig. 2.3. Typically, following the endorsement of the smart contracts by all relevant parties, these contracts are integrated into the blockchain as program codes. Subsequent to their propagation through the peer-to-peer (P2P) network and to their validation by blockchain nodes, they are recorded within the blockchain system. A smart contract is composed of a set of pre-defined states and transition rules. It includes specific scenarios that initiate contract execution, such as the occurrence of a designated time or event, along with predetermined responses to each scenario. The blockchain system actively monitors the real-time status of these smart contracts [21]. Upon the fulfillment of certain predefined conditions – the triggers for contract execution – the blockchain initiates the execution of the contract as per the stipulated terms.

The lifecycle of a smart contract comprises creation, deployment, execution, and completion. Initially, the contract is collaboratively developed by stakeholders, lawyers, and software engineers, and then translated from natural language into a computer-readable format. Upon validation, it is deployed on a blockchain platform, with involved parties' digital assets being locked. The execution phase involves constant monitoring for triggering conditions, leading to automated execution of transactions, recorded on the blockchain. The process concludes with the contract's completion, where the updated states and transactions are recorded, and digital assets are transferred accordingly.

2.1.4 The Ethereum Blockchain Network

Ethereum represents a pivotal innovation in the blockchain domain, introducing programmability into the blockchain universe through its smart contract capabilities. Conceived by Vitalik Buterin and launched in 2015, Ethereum has transcended its initial role as a cryptocurrency platform to become a foundational technology for decentralized applications (dApps). Unlike its precursor, Bitcoin, which is primarily a digital currency, Ethereum's hallmark is its Ethereum Virtual Machine (EVM), an abstracted, global computer that allows developers to write and deploy smart contracts - self-executing contracts with the terms of the agreement between buyer and seller being directly written into lines of code. These contracts run on the blockchain, offering a level of transparency, security, and efficiency previously unattainable with traditional contracts. Ethereum's native cryptocurrency, Ether (ETH), serves a dual purpose: it acts as a digital currency and is used to compensate participants who perform computations and validate transactions on the network, thus maintaining the ecosystem's security and efficiency.

The concept of Ethereum was proposed in late 2013 by Vitalik Buterin, a programmer and co-founder of Bitcoin Magazine, who was intrigued by Bitcoin but felt its potential was limited by its lack of a scripting language for application development. The Ethereum white paper was published in 2014, laying out the vision for a blockchain that could support not only a digital currency but also a wider range of decentralized applications. In 2014, Ethereum conducted a public sale of Ether, raising over $18 million, which was one of the largest crowdfunding events at the time and helped fund the project's development. The Ethereum network officially went live on July 30, 2015, marking the birth of the world's programmable blockchain. Since its launch, Ethereum has undergone several upgrades, known as "hard forks," aimed at improving the platform's scalability, security, and functionality. Notable upgrades include Homestead, Metropolis, and Constantinople. Each upgrade has brought Ethereum closer to its envisioned state as a decentralized platform that enables developers to create applications that run exactly as programmed without downtime, fraud, or interference. The most significant ongoing development in Ethereum's history is the transition from a Proof of Work (PoW) consensus mechanism to Proof of Stake (PoS) through the Ethereum 2.0 upgrade, also known as Serenity. This transition aims to address scalability and energy consumption issues associated with PoW. Ethereum's versatile platform has enabled a myriad of use cases beyond simple transactions, such as Decentralized Finance (DeFi), Non-Fungible Tokens (NFTs), Decentralized Autonomous Organizations (DAOs).

At the heart of Ethereum's architecture are smart contracts, which are programs that automatically execute the terms of a contract when predetermined conditions are met. These contracts are stored on the blockchain, making them tamper-proof and transparent. Smart contracts can be used for a wide range of applications, from simple transactions to complex decentralized applications. The EVM is the runtime environment for smart contracts in Ethereum. It is completely isolated, making it a secure environment for executing code. Every node in the Ethereum network runs an EVM instance, which allows them to agree on executing the same instructions.

2.1.5 The Hyperledger Blockchain Framework

Hyperledger is an umbrella project of open-source blockchains and related tools, started in December 2015 by the Linux Foundation, and has received contributions from IBM, Intel, and other big industry players. Unlike Ethereum, which is public and permissionless, Hyperledger focuses on permissioned blockchain networks for enterprises, emphasizing privacy, scalability, and security.

Hyperledger was launched with the aim of supporting collaborative development of blockchain-based distributed ledgers. With no native cryptocurrency, Hyperledger provides a neutral, open-source infrastructure to support various industry applications, from finance and banking to healthcare and supply chains.The project has grown significantly since its inception, incorporating multiple frameworks and tools designed to facilitate the development of blockchain applications. Some of the key frameworks under the Hyperledger umbrella include Hyperledger Fabric, Sawtooth, Besu, and Indy.

Hyperledger projects are designed with privacy and confidentiality in their core, offering features like channels and private data collections that enable transactions to be private between relevant parties. Its permissioned nature allows for more scalable and high-performance solutions compared to public blockchains, as it can manage consensus more efficiently and process a higher volume of transactions. However, the flexibility and power of Hyperledger come with a steep learning curve. Setting up and managing a Hyperledger blockchain can be complex, requiring a good understanding of the platform and its components.

The key components of Hyperledger's architecture include:

- **Ledger:** The Hyperledger Fabric maintains a ledger for recording transactions. The ledger consists of two parts: the world state and the transaction log. The world state represents the current state of the ledger, while the transaction log records all transactions that have resulted in the current world state.

- **Smart Contracts (Chaincode):** In Hyperledger Fabric, business logic is encapsulated in smart contracts, known as chaincode. Chaincode is written in general-purpose programming languages like Go, Java, or Node.js.

- **Peer Nodes:** The network is made up of peer nodes, which execute Chaincode, access ledger data, endorse transactions, and interface with applications. Peers can be endorsers, committers, or both, depending on their role in the transaction flow.

- **Ordering Service:** The ordering service is a component that batches transactions into blocks and delivers them to peer nodes for validation and commitment. It ensures consistency and finality of the ledger.

- **Channels:** Fabric introduces the concept of channels, allowing a group of participants to create a separate ledger of transactions. This feature supports privacy and confidentiality, as transactions on a channel are only visible to its participants.
- **Membership Service Provider (MSP):** The MSP manages identity and authentication of participants, enforcing access controls in the network. It enables the creation of a permissioned network, where participants have known identities.

Unlike traditional blockchain systems that use a single consensus model, Hyperledger Fabric employs a pluggable consensus mechanism. This allows the network to choose the consensus protocol that best suits its specific needs. The consensus process in Fabric is broken down into three phases, Endorsement, Ordering and Validation.

In the Endorsement phase, transaction execution is performed by endorsers, ensuring that transactions are endorsed according to a policy defined by the network (e.g., endorsed by a specific number or set of peers). Then at the Ordering phase, ordering service batches endorsed transactions into blocks, ensuring that transactions are in a consistent order across the network. At the final phase, peer nodes validate transactions against the endorsement policy and the current state of the ledger, ensuring that transactions are consistent and not in conflict with each other.

2.1.6 Self-Sovereign Identity

The problem of identity management has been a major concern for the scientific community for several years. In particular, the combination of identity management with the full implementation of the GDPR has raised new problems about how to confirm the identity of users without sharing unnecessary information. In this regard, the use of Self Sovereign Identities has been gaining momentum in recent years as a solution that allows for keeping identities secure while proving them to verifiers without revealing unnecessary details. The problem of managing electronic identities is a structural issue that stems from the initial conception of the World Wide Web. The Internet's addressing system is based on the identification of physical network endpoints (computer machines) and not people. Therefore, the Internet cannot uniquely identify users and the identification process is handled by websites and applications. The absence of a secure, portable, and user-controlled identity has dire repercussions. It signifies that a person's identity and personal information exist only within the context of each website or application that he or she uses [22]. This has a significant impact on both the security issue and the increased costs required to set up and maintain multiple identity management mechanisms.

The evolution of identity management mechanisms tries to meet 3 main requirements: (i) security, by ensuring that the relevant information must be protected from unintended disclosure, (ii) control, by ensuring that only the owner is the one who controls who can access it and for what purpose, and (iii) portability, in the sense that the user is not tied to a single vendor.

There are four stages in the development path of digital identities. Centralized identity is the first stage where identities are owned and controlled by a single entity and therefore users don't own their identity record. At a more complex level, it can allow different services to securely share user details without prior knowledge. Federated identity is the second stage of evolvement that answers some of the problems of centralization. Even at the simplest implementation it gives a certain degree of portability compared to the previous approach. For example, a user is able to login into one service using credentials of another. At a more complex level it can allow different services to share details about the user. The User-Centric identity is most frequently manifested in the form of independent personal data stores at one end of the spectrum and large social networks at the other end [23]. However, the entire spectrum still relies on the user selecting an individual identity provider and agreeing to their often one-sided adhesion contracts. The latest advancement on identities is Self - Sovereign Identity (SSI) which gives users the ability to exercise control over the information that is associated with their digital identities and the capability to demonstrate to websites, services, and applications across the internet that they are who they say they are. The SSI is defined as the digital movement, which states that individuals should own and manage their digital identity without the need for intermediate authorities. In real life, the proof for various things is provided by showing a certificate, a card or a piece of paper that is kept by the identity holder in a safe place or a wallet. SSI applications essentially digitize this whole process.

In the sections that follows, we present the main components of SSI, along with a novel SSI management framework that can expand the proposed HMAC system.

2.1.7 The distributed DDI ecosystem

Distributed Self-Sovereign Identity (SSI) is a digital movement that enables individuals and organizations to have sole ownership and control over their identity and personal data, allowing them to interact in the digital world with the same level of trust and freedom as they do offline. SSI enables entities to demonstrate control over their identifiers without the use of intermediaries and provides the flexibility to choose which information or claims are disclosed to third parties. SSI seeks to address flaws in existing procedures that fail to protect personal and sensitive information from unauthorized access or data breaches. Leveraging the Distributed Ledger Technology (DLT), SSI can improve security, trust, and efficiency while supporting the privacy of user data, which is crucial for businesses managing customer and employee authorizations and the increasing demand for inter-domain communications and cross-organizational authorization.

A distributed SSI system is based on blockchain technology, which provides a decentralized and secure platform for storing and managing identity data. These blockchain-based systems enable users to maintain control over their personal information by creating and managing their own digital identities, without requiring a central authority to verify or authenticate their identities by keeping a centralized record for all users. Instead, all data are distributed across a network of nodes, with each node maintaining its own copy of the blockchain. Because there is no central point of failure, it is difficult for adversaries to compromise the system.

In the SSI ecosystem, specific roles of basic actors are defined that interconnect in order to establish trust. An overview of the SSI's core entities and their relationships are presented in Fig. 2.4, where a logical triangle of trust is shaped among them. Issuer, Holder, and Verifier are the three primary entities of the system where each role has specific responsibilities and functions to ensure the security, privacy, and efficiency of the system. The Issuer is responsible for creating and verifying VCs, the Holder can create and manage the digital identity using decentralized identifiers (DIDs) and store their VCs in a secure and private manner and the Verifier is the entity that request the VCs from the Holder, which are verified by the Issuer. Additionally, a Verifiable Data Registry (VDR) is required for the ecosystem to function. VDR is accountable for creating and validating identifiers, keys, and other pertinent data required and used by verifiable credentials, such as credential schemas, revocation registries, issuer public keys, etc.

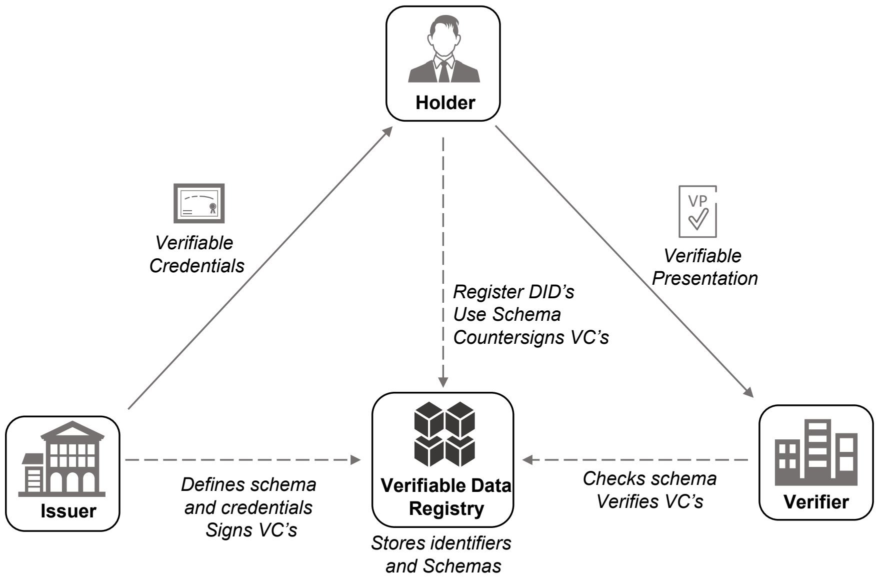


Figure 2.4: SSI triangle

In the process of Self-Sovereign Identity (SSI), the Holder initiates a request for a credential from the Issuer, and then presents it to the Verifier. The Verifier then accepts the credential and proceeds to verify it through the Verifiable Data Registry (VDR). Upon completion of the verification process, the Verifier is able to place trust in the verifiable credential, which has been signed by the Issuer, just as it would with the Issuer itself.

A distributed SSI system can reduce the cost and complexity of identity management. Since there is no central authority managing the system, there are no intermediaries to pay and no complicated authentication procedures to navigate. This improves the system's efficacy and cost-effectiveness, and can also aid in preventing fraud and identity theft. By leveraging blockchain technology and SSI-specific solutions such as DIDs, VCs, and DLT, these systems enable individuals and organizations to have greater control over their personal data, while simultaneously reducing the risks of data breaches and unauthorized access.

2.1.8 SSI Main Components

Verifiable Credentials (VCs) and *Verifiable Presentations (VPs)* are crucial components of Distributed SSI systems. VCs are digital records containing information related to identifying the subject of the credential (e.g., a person or organization), the issuing authority, and/or the type of credential, which have been previously validated by a trusted party. VPs are digital artifacts that enable subjects to selectively disclose VCs to others based on the transaction context and their personal preferences. In a decentralized SSI system, VCs are stored on a DLT, such as a blockchain, which ensures that they cannot be altered or tampered with, providing a secure method of verifying a subject's identity or other attributes without the need for intermediaries or centralized authorities. The holders of verifiable credentials are able to create verifiable presentations, which they can then share with verifiers to prove that they possess verifiable credentials with specified attributes. A subject may wish to share proofs of eligibility for accessing specific data without disclosing unrelated personal information. In contrast to traditional identity management systems, VCs, allow for the flexible and user-centric sharing of information. VCs and VPs are managed using a combination of blockchain technology and cryptographic protocols in a distributed SSI system. Digital Identity Decentralized Identifiers (DIDs) are used to uniquely identify individuals and organizations, whereas Verifiable Data Registry (VDR) is used to store and manage VCs. To verify the authenticity and integrity of VCs and VPs, digital signatures and Zero-Knowledge Proofs (ZKPs) are utilized. Note, however, that verifiability of a credential does not imply that the validity of claims encoded within can be evaluated; however, the issuer can add values to the evidence property to help the verifier apply their business logic to determine whether the claims have sufficient veracity for their purposes.

Verifiable Data Registries (VDR) are decentralized data storage systems that can be used to store VC's that represent various attributes of an individual identity (e.g. name, age, role and access attributes). Through the application of cryptographic techniques, these properties can be validated by third parties of trust.

One of the primary benefits of adopting VDRs in a system of self-sovereign identity is that they enable complete control over personal data. Instead of relying on centralized identity providers to store and manage their data, all data can be stored in a safe and decentralized VDR, with access granted on an case-by-case basis. This not only provides users with more control over their data, but also minimizes the likelihood of data breaches and identity theft. Moreover, the use of VDRs can facilitate increased compatibility between various identification systems. Since VDRs are based on open standards, they may be used to store and exchange verified credentials across many identity systems, allowing individuals to access services and apps across several platforms with ease.

To ensure the security and integrity of VDRs, a number of essential elements are required. They include robust encryption and cryptographic techniques, as well as mechanisms for protecting data privacy and preventing illegal access. In addition, VDRs should be highly available and robust, with redundant storage and backup systems to ensure that data is constantly accessible and recoverable in the event of a breakdown.

Decentralized identifiers (DIDs) are a new type of identifier that enables digital identity to be self-owned and controlled by the user. DIDs are unique identifiers that are anchored on decentralized networks like blockchain or distributed ledgers, making them tamper-proof, immutable, and independent of centralized identity providers. In distributed self-sovereign identity (SSI), DIDs are used to provide a user-centric approach to identity management. This means that individuals have complete control over their identity and personal data, and they can choose to selectively disclose their identity attributes to others without relying on a third party. DIDs enable the creation of verifiable credentials, which are digital representations of identity attributes that can be cryptographically signed by the issuer and verified by the recipient. This creates a secure and transparent way to share personal data and verify identity claims without the need for a centralized intermediary. DIDs also provide privacy by design as they do not rely on a centralized identity provider or a single point of failure. Instead, DIDs use peer-to-peer networks and decentralized storage systems to ensure that personal data is not held in a single location, reducing the risk of data breaches or identity theft.

2.2 Related Work

2.2.1 Fined-grained Access Control Mechanisms

Fine-grained access control mechanisms play a pivotal role in multi-authority and multi-domain environments, where security and privacy concerns are paramount. These environments are characterized by their complexity, involving multiple stakeholders with varying degrees of authority and distinct domain-specific security requirements. Fine-grained access control is designed to address these challenges by enabling precise specification and enforcement of access policies at a highly detailed level. This approach allows for the delineation of access rights not just by user role or group, but also by specific actions, resources, and context conditions, thus providing a more nuanced control over sensitive information.

Central to the implementation of fine-grained access control in such heterogeneous environments is the Attribute-Based Access Control (ABAC) model. ABAC utilizes attributes related to users, resources, and the environment to make access decisions, allowing for dynamic policy enforcement that can adapt to changes in user roles, resource classifications, or environmental conditions. Furthermore, cryptographic techniques such as Ciphertext-

Policy Attribute-Based Encryption (CP-ABE) are increasingly integrated to ensure secure and efficient access control in distributed settings. These mechanisms enable organizations to enforce access policies that reflect their complex operational and security requirements, enhancing data protection and compliance across diverse domains. Through the adoption of fine-grained access control, multi-authority and multi-domain environments can achieve a balance between flexibility and security, ensuring that only authorized entities can access the specific resources they need, under appropriate conditions.

Several frameworks have been proposed in the literature for fine-grain authorized access to centralized sensitive resources, for example healthcare data [24, 25, 26, 27, 28] and healthcare records [29, 30, 31, 32, 33, 34, 35, 36, 37]. Studies related to secure access for IoT-enabled smart healthcare devices are limited. For example, in [38] the authors propose a novel access control architecture that provides fine-grain access to IoT medical devices. In [39] a fine-grain data access control mechanism is combined with the implementation of fog computing to provide high-level privacy for IoT medical device applications. A cloud-based fine-grain health information access control framework for lightweight IoT devices with data dynamics auditing and attribute revocation functions is provided in [40].

Sharing electronic health records among healthcare stakeholders is essential since typically more than one healthcare provider is involved in a patient's treatment [41, 42]. Although the above approaches may provide elegant solutions for granular access control, they are not concerned with forensics characteristics such as provenance, which are required in distributed environments. Another line of research addresses authorization for retrieving information through cryptographic techniques such as searchable encryption, e.g. [43, 44, 45].

Recently blockchain technology has been proposed as a mechanism for granular access control to information. Most of the available studies in the literature focus on the development of blockchain-enabled authorization frameworks of decentralized record management systems for handling electronic records [46] but they do not cover access to IoT devices. For example, in [47] a medical data access and permission management framework is provided which allows specific authorizations. A patient-centric healthcare data management system based on blockchain technology is proposed in [48] which takes into account authorization access to individual patient health data. In [49], a system for identity and access management using blockchain technology to support authorization of entities is proposed. A blockchain-based conceptual framework with secure cryptographic user management properties and hierarchical authorizations to healthcare medical records is proposed in [50]. A blockchain-based framework using users' cryptographic keys and identities is proposed in [51], where authorized queuing requests may be performed. In [52], a cloud storage scheme to manage and share personal medical data with authorized access control based on blockchain is proposed. Finally, a blockchain-based electronic medical records architecture with different levels of granularity for authorization is proposed in [53].

During the last few years, several research attempts have tried to provide fine-grained access control in the dynamic multi-authority and multi-domain setting, while maintaining interoperability [54]. Some of these works have focused on decentralization and privacy-preserving encryption [55, 56, 57]. Such targeted solutions enable adequate compatibility and fine-grained access control; however, they fail to combine credentials issued by independent authorities. Other solutions rely on privacy-preserving encryption, such as ABAC. For example, Ref. [58] proposes a decentralized MA-ABAC (DMA-ABAC) scheme for multi-domain healthcare ecosystems. In this work, even though authorities can independently control their security settings and enforce cross-domain policies, the lack of a mechanism obliging users to use the system for accessing the data, in combination with the absence of global verifiability, leads to strong trust assumptions. In [59], an ABAC solution is designed for the shared multi-owner setting, assuming a distributed setting with multiple authorities. The authorities own pieces of data and may issue attribute keys that users may combine to access data belonging to different authorities. Even though the proposed solution strengthens the restrictions for accessing data, it falls short on addressing the challenge of inter- and cross-domain policy enforcement, i.e., dynamically defining access policies both to control access for all authorities within a domain or for all authorities belonging in different domains. Many works try to solve this problem by using Multi-Authority Ciphertext Policy Attribute-Based Encryption (MA-CP-ABE).

In [60], a MA-CP-ABE scheme is proposed that supports range policy, which, however, maintains the need for a trusted central authority. In [61], the authors adopt the idea of hybrid encryption to reduce the computational overhead of data encryption, using a symmetric key to encrypt data and a MA-CP-ABE mechanism to encrypt the symmetric key. The authors in [62], also propose a MA-CP-ABE scheme combined with outsourced decryption and zero-knowledge proof for the Internet of mobile things. In [63], another MA-CP-ABE scheme is proposed as suitable for IoT applications and devices with low computational capabilities. Decryption operations are outsourced to fog for efficiency reasons, but no policy management is supported. The authors of [64] introduce a token-based access control scheme that uses smart contracts and blockchain to generate decryption keys according to verified user attributes. In [65], Elliptic Curve Cryptography (ECC) and MA-CP-ABE are combined to create an access control system that supports the setting of multiple authorities, but the addition or removal of authorities remains inefficient. In [66], the authors propose a privacy-preserving MA-CP-ABE scheme for blockchain-based applications in the supply chain. This model achieves fine-grained access control and versatile authorization and also protects the user's private key from leakage even when some attribute authorities fail. However, in most of the above solutions, trust expectations about global transaction verifiability persist. Strong trust assumptions are required to ensure that all access transactions to encrypted data are immutably logged and may be globally verified by all entities. In [67], the authors also suggest an access model based on CP-ABE for IoT in healthcare, and although they achieve the collaboration of independent authorities, they do not solve the need for inter- and cross-domain policy management. In [68], an Attribute-based Signcryption (ABSC) scheme is proposed that relies on a central certificate authority to verify the attribute authorities and thus maintains strong trust assumptions.

Other works use hierarchical attribute-based access models, e.g., [69, 70, 71, 72, 73]. For example, in [72], the author present a hierarchical Multi-Dimensional Access Control (MD-AC) model for the authorization of multiple participants in the cloud. Furthermore,in [73], the authors present a Cross-Domain Access Control (CD-AC) model based on a trusted third party (TTP) and an attribute mapping center (MC) that incorporates CP-ABE to provide granular access to users. Both works use the cloud for efficiency reasons, assuming strong trust. In general, although hierarchical attribute-based access models are flexible and scalable, they are not suitable for multi-authority, multi-domain environments, where roles may not have a global and strict hierarchy.

The current state-of-the-art leverages blockchain technology to provide a variety of fully autonomous and hybrid solutions [74, 75]. Although blockchain technology is more difficult to administer and may introduce efficiency issues, it provides, by design, global verifiability of data access transactions. An immutable ledger can ensure the integrity of transactions and data while also enforcing trust among multiple untrustworthy parties [14]. For example, in [76], the authors propose a fine-grained access control scheme for transportation ecosystems based on blockchain, that embraces the multi-owner setting with the use of CP-ABE. The system provides global verifiability and data integrity, but cannot support dynamic joins and leaves of nodes, or dynamic policy management. Hybrid solutions move some of the services previously supported by cloud providers to the blockchain. While this approach resolves some of the issues (e.g., data integrity), others problems, such as strong trust assumptions for the cloud operator, remain [77, 78, 79]. Although autonomous solutions are entirely based on the Blockchain, they are still limited by the level of efficiency that can be supported [1, 57, 76, 80]. In addition, the scalability of fine-grained access control mechanisms in such complex scenarios also necessitates the deployment of decentralized identity and access management systems. These systems facilitate interoperability and trust across different domains and authorities without compromising on security or privacy.

In conclusion, fine-grained access control mechanisms are crucial for securing multi-authority and multi-domain environments. Through the judicious application of ABAC models, cryptographic techniques and decentralized systems, these mechanisms can offer robust, dynamic, and scalable solutions to meet the intricate security demands of such ecosystems. Blockchain technology, offers a promising solution by providing a tamper-proof and transparent platform that can handle identities and access permissions. Through smart contracts, blockchain can automate the enforcement of access policies, reducing the administrative overhead and enhancing the responsiveness of access control systems. As technologies evolve, further advancements in access control strategies will continue to strengthen the security and privacy of information in complex and interconnected digital landscapes.

2.2.2 Distributed Trust Management Mechanisms for Complex Ecosystems

Distributed trust management mechanisms are another essential component for upholding the integrity, security, and operability of complex ecosystems, particularly in scenarios that span multiple domains requiring robust certification management. These environments, which can range from digital economies and decentralized platforms to intricate IoT frameworks, are defined by their extensive diversity, vast scale, and the imperative for seamless interoperability amongst a plethora of stakeholders. A fundamental challenge within such settings is to establish and maintain trust without the reliance on centralized authorities. Distributed trust management mechanisms cater to this need by fostering a decentralized model for trust verification and decision-making, with a significant focus on certification management and cross-domain certification interoperability.

Certificate-based user authentication stands as a cornerstone for secure communications and access control in distributed systems, especially when it comes to complex ecosystems involving numerous stakeholders. This method of authentication leverages digital certificates, which are issued by trusted Certificate Authorities (CAs), to verify the identity of users and devices. The essence of certificate-based authentication lies in its ability to provide a secure and scalable means of verifying identities across diverse systems and networks, eliminating the reliance on traditional, less secure methods such as passwords. In the existing literature, several scholars have contributed to the development of architectures that integrate Public Key Infrastructure (PKI) with the aim of enhancing security in various contexts. For instance, the authors in [81] propose the PKIoT architecture, which facilitates secure Internet of Things (IoT) transactions while also reducing energy consumption. This architecture is distinguished by its introduction of compact certificates, which are specifically designed to support efficient communication within IoT environments. In a related vein, the study presented in [82] extends traditional role-based access control mechanisms by incorporating PKI for role assignment, thus enhancing the security framework within organizational settings. Additionally, in [83], the authors develop a distributed security policy management architecture that leverages PKI to address the challenges of decentralized access control and authorization policies. This architecture is particularly noteworthy for its ability to enforce security policies across disparate systems in a cohesive and secure manner.

For stakeholders within these ecosystems, the need to maintain an individual infrastructure that issues certificates for their users is paramount. This infrastructure, typically revolving around a robust PKI, enables the generation, distribution, and management of digital certificates. By having their own PKI, organizations can tailor the certificate issuance process to meet specific security policies and compliance requirements, thus ensuring a higher level of control over access to resources and data. Furthermore, an individualized PKI infrastructure facilitates the swift revocation of certificates when necessary, enhancing the system's ability to respond to security breaches or compromised credentials.

However, the efficacy of certificate-based user authentication in multi-stakeholder environments hinges not only on the ability of entities to issue their own certificates but also on the establishment of a mechanism that provides trust for certificates issued by different authorities. This cross-certification or trust bridging is crucial for enabling secure interactions and interoperability across domains and organizational boundaries. Without such mechanisms, the recognition and validation of certificates issued by external CAs would be fraught with complexity and security risks, undermining the seamless operation of distributed systems.

To overcome this challenge, numerous studies have proposed certificate-free solutions. For example, in [84], the authors introduce a system designed to facilitate the sharing of patient data between cooperative organizations while preserving privacy, which is based on roles and signatures. In [85], an authenticated asymmetric group key agreement protocol is presented, featuring a self-certification mechanism for keys that allows participants to verify the correctness of the group keys. Furthermore, the researchers in [86] propose a cloud-based architecture for cross-domain data sharing. Although these solutions mark progress towards establishing cross-domain trust, the absence of certificates may compromise trust among participants and pose scalability challenges for trust management mechanisms. There remains a pressing need for trust management systems that provide interoperability among various authorities and can fulfill the demands of real-world applications.

To address this challenge, blockchain technology is often presented in the literature as a promising solution to improve trust in certificate-based user authentication. By storing certificate revocation lists (CRLs) or issuing certificates on a blockchain, stakeholders can leverage its immutable and transparent nature to ensure the integrity and verifiability of certificate statuses across different authorities. Blockchain-based systems can also automate the external certificate validation process, reducing operational overhead and improving the efficiency of trust management practices. For example, the authors in [87], leverage a consortium blockchain to achieve cross-domain authentication and privacy protection for IoT devices with limited computing capabilities. For instance, the authors in [87], utilize a consortium blockchain to facilitate cross-domain authentication and privacy protection for IoT devices with limited computing capabilities. In [88], a blockchain-based PKI identity management system is proposed, utilizing smart contracts to streamline certificate management and enhance cross-domain authentication efficiency. Moreover, in [89], a flexible blockchain-assisted secure authentication mechanism is presented for cross-domain industrial IoT applications. This solution allows devices from different administrative domains to authenticate each other while preserving anonymity. Although these solutions address certain challenges, they still exhibit limitations within complex real-world ecosystems, with ongoing concerns related to scalability, efficiency, and security.

In conclusion, distributed trust management mechanisms, with a focus on certification management and cross-domain certification interoperability, are indispensable for the secure and efficient functioning of complex ecosystems. Using decentralized technologies and establishing common standards, these mechanisms provide a foundation for trust that transcends domain boundaries without the reliance on central authorities. As ecosystems continue to evolve and interconnect, the development and refinement of distributed trust management strategies will remain a critical area of focus, ensuring the security and prosperity of digital and physical networks alike.

2.2.3 Privacy Preserving Encryption for MA-MD environments

In the digital age, the need for privacy-preserving encryption in multi-authority and multi-domain environments has become more critical than ever. As organizations and systems increasingly interact across various domains and jurisdictions, ensuring the confidentiality and integrity of sensitive information while maintaining user privacy is crucial.

In multi-authority and multi-domain environments, data is often shared across different entities, each with its own governance, policies, and security protocols. The complexity of these interactions exacerbates the risk of data breaches, unauthorized access, and privacy violations. Privacy-preserving encryption addresses these concerns by enabling secure data sharing and processing without compromising the privacy of the underlying data. It ensures that sensitive information remains encrypted during transmission and storage, only accessible to authorized parties with the decryption keys.

Several studies have proposed the implementation of privacy-preserving encryption, each specifically tailored to the unique requirements of multi-authority and multi-domain environments. Some of these studies have focused on homomorphic encryption schemes. The first fully homomorphic encryption scheme was introduced by G. Gentry in [90], yet advancements in this field have been modest. Homomorphic encryption enables computations on encrypted data, producing an encrypted result that, once decrypted, matches the result of operations performed on the plaintext. This allows entities to perform data analysis and processing without accessing the raw data, thereby preserving user privacy. Although these schemes are valuable, they impose significant restrictions on the range of actions that can be performed on the data [91]. Current homomorphic encryption schemes remain impractical for most real-time or large-scale applications due to their substantial computational overhead. To achieve efficient trust management while controlling access to data, these schemes should be integrated with other cryptographic methods.

Other studies, propose Attribute-Based Encryption which is a flexible encryption scheme where access to encrypted data is based on attributes (e.g., user roles, location, or time) rather than the identity of the user. In ABE, policies can be integrated into the encryption process, ensuring that only users with matching attributes can decrypt the data. This method is ideal for multi-authority environments, as it allows for fine-grained access control across different domains without the need for a central authority.

ABE schemes can be broadly classified into two categories: Key-Policy Attribute-Based Encryption (KP-ABE) and Ciphertext-Policy Attribute-Based Encryption (CP-ABE). In KP-ABE, the encryptor defines a set of attributes for the data, and the keys issued to users are associated with access policies. If a user's policy matches the data's attributes, they can decrypt the information. Conversely, CP-ABE reverses this approach by associating access policies with the data itself, and users are given keys based on their attributes. If the user's attributes satisfy the data's policy, decryption is permitted. This flexibility allows organizations to implement detailed access control mechanisms that reflect their operational and security requirements accurately.

In the first category, several researchers have proposed KP-ABE schemes tailored to meet the needs of MA and MD ecosystems. For example, in [92], the authors have developed a scheme that incorporates a cryptographic reverse firewall, designed to facilitate secure and efficient data sharing in cloud environments. This method effectively addresses the challenges of decentralized attribute management. Similarly, [93] introduced a decentralized KP-ABE scheme that uses blockchain technology for smart grid applications, which mitigates trust issues in distributed authorization and reduces computational costs associated with traditional ABE schemes, while utilizing blockchain for enhanced security and transparency. However, KP-ABE schemes involve complex user certificate verification and secret key distribution processes managed by a single attribute authority, which can create performance bottlenecks and place an undue burden on resource-limited devices. This is particularly problematic in distributed multi-authority systems where efficiency and scalability are critical. Moreover, these schemes often lack adaptive security in standard models, which restricts their flexibility and applicability in dynamic environments.

Other studies, focused on CP-ABE schemes, which is a form of encryption that enables access control over encrypted data through the use of policies and attributes. In CP-ABE schemes, the encryptor defines a policy, typically expressed as a logical combination of attributes, which determines the decryption capabilities. Each user is issued a set of attributes embedded in their secret keys. When a user attempts to decrypt a ciphertext, the decryption process checks whether the user's attributes satisfy the policy associated with the ciphertext. If the policy is satisfied, the decryption succeeds; otherwise, it fails. Recent works have introduced significant improvements in CP-ABE schemes, particularly in terms of security and efficiency. For instance in [94], the authors developed a CP-ABE scheme based on lattice LWE, focusing on security analysis to establish its robustness against common cryptographic attacks. In [95] the authors propose an efficient and revocable CP-ABE schemes with outsourcing decryption, specifically designed for IoT applications, to reduce the computational load on devices. However, revoking attributes in CP-ABE is problematic because it often requires re-encrypting the existing data or re-distributing new keys to all affected users. This process can be computationally expensive and difficult to scale in large systems.

A particular CP-ABE scheme seems to stand out as the most suitable for MA and

MD environments. Multi-authority ciphertext-policy Attribute-Based Encryption (MA-CP-ABE), unlike traditional encryption methods, which rely on a single authority to manage encryption and decryption keys, allow multiple authorities to govern access control. MA-CP-ABE was initially proposed in [96] as an application of Attribute-Based Encryption in which any party can become an authority with no global coordination requirements. However, the scheme required a trusted central authority to collect all master private keys from all Attribute Authorities (AA) to compute the collective secret terms for system initialization. MA-CP-ABE was later extended by [97] introducing fully decentralized CP-ABE systems for both composite-order and prime-order groups by utilizing the user's Global Identifier (GID) in the key to resist collusion attempts. Several other works have extended the features of the scheme to allow fine-grained data access with attribute revocation for cloud data storage [98], improved efficiency [99] and storage space saving by using hierarchical attributes to compress redundant ciphertext information ([100, 101]).

With a MA-CP-ABE scheme, multiple authorities agree on a set of global parameters GP and, based on these parameters, each authority X generates a public/secret key pair PK_X, SK_X. Data m can then be encrypted based on a mutually agreed access policy $\mathbb{P}$ (in the form of a matrix), using the public keys of all the authorities, i.e., $ct = Enc(m, \mathbb{P}, \mathrm{GP}, \{PK\})$. With MA-CP-ABE schemes, any party can become an authority, and there is no requirement for a global root authority. More importantly, users can combine attributes issued by different authorities, provided that each user has a unique global identity GID. Any authority X may issue to any user U, attribute keys for an attribute $attr$, using its private key, the global parameters, and the user identifier:

$$K_{U,attr} = KeyGen(\mathrm{GID_U}, \mathrm{GP}, attr, SK_X).$$

Finally, users can combine their attribute keys, issued by multiple authorities, to decrypt an ABE-encrypted ciphertext ct, provided that their set of attributes satisfies an access rule within $\mathbb{P}$, i.e.,: $m = Dec(ct, \mathrm{GP}, \{K_{U,attr}\})$. Although the scheme enables a combination of attributes issued by different authorities, it remains collision-resistant, meaning that different users cannot combine their attributes, since each attribute key is assigned to a different GID. An example of MA-CP-ABE from the medical ecosystem is illustrated in Figure 2.5.

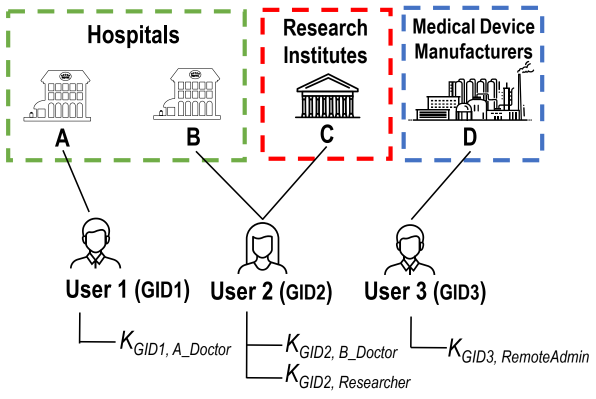


Figure 2.5: MA-ABE attribute binding and key generation

Several organizations belonging to different domains may agree on inter-domain or cross-domain access policies. A domain-wise policy for hospitals may be, for example, to allow access to patient health for doctors at any hospital, if the patient is under emergency treatment. A cross-domain policy for the hospital domain may be to allow access to anonymized data to researchers, or to access the configuration data of medical devices to authorized manufacturer admins. Users may be given attribute keys from different organizations (authorities) and combine them since each attribute key is linked to the global identifier of a user. However, keys issued to different users cannot be combined.

However, implementing the MA-CP-ABE scheme presents several challenges, particularly in creating an architecture that supports efficient decentralization and distribution of decryption functionality. One of the primary challenges is achieving interoperability among multiple authorities without compromising security or privacy. Each authority operates independently, issuing keys based on its own set of policies and attributes. Ensuring that these disparate systems work seamlessly requires a robust framework that can handle attribute and policy management across different domains. Moreover, scalability becomes a significant concern as the number of authorities and users grows, necessitating efficient key distribution mechanisms and minimal overhead in decryption processes.

Another critical challenge in deploying MA-CP-ABE systems is maintaining privacy and security in a decentralized setting. In addition, the system must be resilient to various security threats, including compromise of authority and insider threats. Developing an architecture that addresses these challenges while ensuring an efficient and practical implementation of MA-CP-ABE could prove difficult.

SECTION II

Hierarchical Multi-blockchain

CHAPTER 3

HIERARCHICAL MULTI-BLOCKCHAIN MODEL

In the evolving landscape of blockchain technology, the integration of access control mechanisms that are both scalable and secure across multiple domains remains a paramount challenge. In this chapter, we present a novel Hierarchical Multi-Blockchain Access Control Model designed to seamlessly manage access within multi-authority and multi-domain environments. This model takes advantage of the decentralized nature of blockchains while introducing a structured hierarchy to improve efficiency, scalability, and security in access control operations. As pointed out in Chapters 1 and 2, traditional centralized access control systems often struggle with transparency and are prone to single points of failure, making them unsuitable for the trustless MA and MD environment. Conversely, existing decentralized solutions typically offer greater transparency but face significant challenges in terms of scalability and complex management in multi-stakeholder scenarios. Our proposed Hierarchical Multi-Blockchain Access Control Model addresses these issues by establishing a tiered access structure, where different blockchains with varying levels of authority and function are integrated under a unified framework. This architecture allows for a clear demarcation of roles and responsibilities, reducing the risk of unauthorized access while enhancing the overall system's ability to scale and adapt to new requirements.

3.1 Requirements

Interoperable data access in MA and MD ecosystems 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 ecosystem stakeholders and the very nature of the data handled (shareble or sensitive data). Clearly, the security and functional requirements should be combined and context-specific, as described below.

Decentralized, interoperable and adaptable trust management. The system must allow all stakeholders to manage trust within their domain in a distributed and self-sustained fashion. For example, it must be possible for each stakeholder 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 within the relevant domain, provided there is consensus among the existing stakeholders, in a flexible but also secure manner.

Fine-grained access control. The system must ensure that all entities/stakeholders 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 and emergency department. In the supply chain ecosystem, a logistic service provider may allow temporal access to a technician for accessing device data. Such temporal access should be easily revocable. Finally, the system must ensure access control that bypasses avoidance mechanisms.

Distributed, efficient, and 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 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 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.

Expressive and decentralized access policies. A mechanism should be used to enforce decentralized policies, and allow mutually untrusted stakeholders to manage their data in an expressive and autonomous way. Colluding stakeholders should not affect the execution of system services. In the event of disagreement on the context of a transaction, a consensus mechanism could be used (typically, a Proof of Stake (PoS)).

Tailored forensics. The system should provide a strong forensics-by-design mechanism, capable of providing 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 device vigilance. 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.

3.2 Hierarchical Multi-blockchain Access Control Model (HMBAC)

At a high level, the goal of HMBAC is to allow users belonging to different stakeholders (authorities) from different domains to have controlled access to data owned by multiple stakeholders. Moreover, the model must support interoperable use of credentials issued by different authorities, while the inter- and cross-domain policy management must be controlled at a domain level. Hierarchical multi-blockchains play a central role in the HMBAC model. The Proxy Blockchain layer answers the first goal by allowing the interoperable use and verification of credentials issued and managed independently by different authorities and domains. Then, the various Domain Blockchains answer the second goal by allowing authorities belonging to different domains to define inter- and cross-domain policies for their data and to manage the authority membership in their domain, without affecting other domains. Following the approach in Malamas et al. [5], we define the HMBAC model by representing the logical relations among the access control components, as shown in Figure 3.1. A user (U) belonging to one or more authorities (Au) (for example, a hospital if it is for the medical ecosystem or a transportation company if it is for supply chain) is assigned attributes from a pool of attributes (UA). One or more authorities may issue attributes to users, which is depicted as the attribute authority (AA) relation in Figure 3.1. Each authority is associated with a single domain (D) and each domain contains multiple Au. A key element in the HMBAC model is the hierarchical multi-blockchain. All domains, and thus all authorities, constitute the proxy blockchain (PBc), i.e., these are the stakeholders for the first layer of the multi-blockchain. Then, at the second layer, various domain blockchains (DBc) can be constructed. Each group of authorities with similar characteristics forms a different DBc (e.g., one DBc is constructed by the hospital stakeholders, while another DBc is constructed by the manufacturers). Objects (OB), representing data or services accessible by users and operated by subjects (S), are encrypted with attribute keys (AK) created by the UA pool. With the terms user attribute authorities (UAA) and subject attributes (SA), we represent the logical connection between users and subjects with the UA pool.

For a user to gain access to encrypted data, three checkpoints must be met. First, an attribute verification function (AVF) executed on the PBc will verify the validity of the user attributes. Then, the authorization function (AF), placed in the DBc, will verify whether the attributes of the presented user are sufficient, based on the relevant inter-domain (IDP) and cross-domain (CDP) policies, to authorize the access request. Finally, the decryption function (DF) also executed in the relevant DBc of the data owner, will partially decrypt the data, which will be fully decrypted by the user, with the proper user attribute keys.

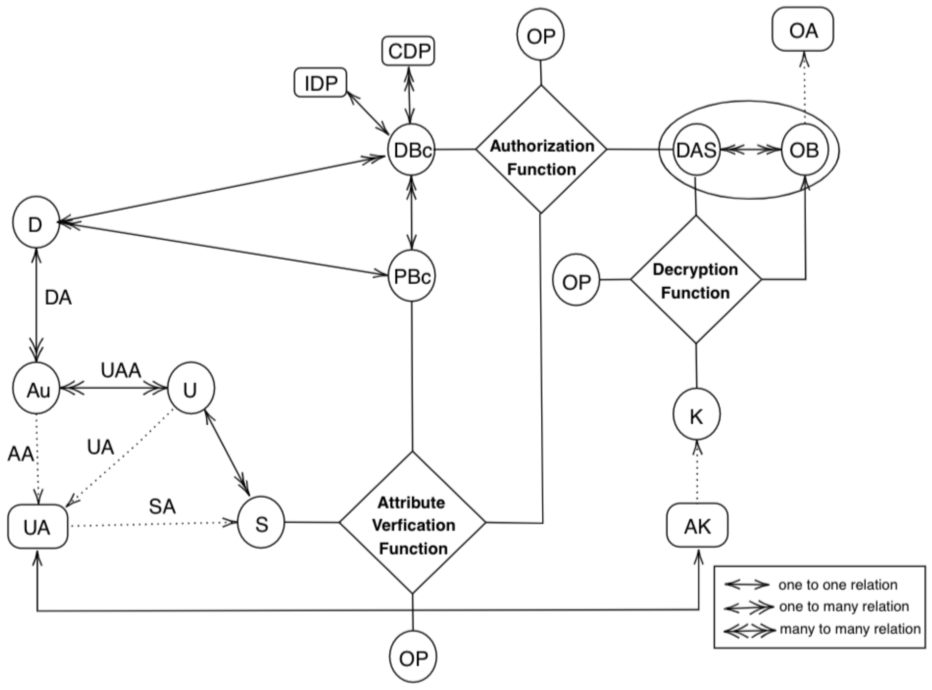


Figure 3.1: HMBAC access model: element sets and relations.

Table 3.1, summarizes the basic sets and functions of the proposed HMBAC model, as well as a formal analysis of the three functions specified for data access and decryption. Users, objects, and keys can be assigned attribute values directly from an attribute function att from a set of values in the range, denoted $Range(att_u)$, $Range(att_{OB})$, $Range(att_K)$, respectively. Users are assigned to multiple authorities (defined by many to many functions $direct_{UAu}$) and also authorities are assigned to one domain (defined by one to many functions $direct_{DAu}$).

Table 3.1: Basic sets and functions of the HMBAC model

Basic Sets and Functions

-U, Au, S, K, D: finite sets of users, authorities, subjects, keys, domains

-UA, OA, AK: finite sets of user, object, and keys attribute functions

- PBc, DBc: fine sets of Proxy and Domain blockchain services

-IDP, CDP: fine sets of inter and cross domain policies

-OB, OP, DAS: fine sets of objects, operations and data services

-$attType : UA = \{set\}$, defines user attributes to be set valued only.

-$attType : AK = \{set\}$, defines keys attributes to be set valued only

Each attribute att_U in UA maps users or authorities to a set of attribute values in $Range(att_U)$. Formally, $att_u : \; U \cup Au \to 2^{Range(att_U)}$

Each attribute att_{OB} in OA maps objects in OB to attribute values. Formally, $att_{OB} : OB \to 2^{Range(att_{OB})}$

Each attribute att_K in AK maps keys in K to attribute values. Formally, $att_K : K \to 2^{Range(att_K)}$

Direct $UAu : \; U \to 2^{Au}$, mapping each user to a set of authorities.

Direct $DAu : \; D \to 2^{Au}$, mapping each domain to a set of authorities.

Effective Attributes of Users, Subjects and Keys

For each attribute att_U in UA, $effectiveAu\ _{att_U} : Au \to 2^{Range(att_U)}$

For each attribute att_U in UA, $effectiveU\ _{att_U} : U \to 2^{Range(att_U)}$

$US : S \to U$, mapping each subject to a user

For each attribute att_U in UA, $effectiveS\ _{att_U} : S \to 2^{Range(att_U)}$, mapping each subject to a set of values for its $effectiveU\ _{att_U}$.

For each attribute att_K in AK, $effectiveK\ _{att_K} : K \to 2^{Range(att_K)}$.

The three functions are defined as follows:

Definition 1. *(Attribute Verification Function) A subject* $\mathrm{s} \in \mathrm{S}$*, is allowed to perform* $\mathrm{op} \in \mathrm{OP_{AVF}}$ *on a service* $\mathrm{sr} \in \mathrm{PBc}$*, if* $\mathrm{effectiveS_{att_v}} \in \mathrm{UA}$*. Formally,* $\mathrm{OP_{AVF}(s : S, sr : PBc) = True}$

Definition 2. *(Authorization Function) A subject* $\mathrm{s} \in \mathrm{S}$*, is allowed to perform* $\mathrm{op} \in \mathrm{OP_{AF}}$ *on a service* $\mathrm{sr} \in \mathrm{DBc}$*, if* $\mathrm{effectiveS_{att_v}}$*, satisfies the policies stated in* $\mathrm{Auth_{DBc}(s : S, sr : IDP \cup CDP)}$*. Formally,* $\mathrm{Auth_{DBc}(s : S, sr : IDP \cup CDP) = True}$

Definition 3. *(Decryption Function) A subject* $\mathrm{s} \in \mathrm{S}$*, is allowed to perform* $\mathrm{op} \in \mathrm{OP_{DF}}$ *on an object* $\mathrm{ob} \in \mathrm{OB}$*, in data access services* $\mathrm{ds} \in \mathrm{DAS}$*, if* $\mathrm{OP_{AVF}(s : S, sr : PBc) \cap Auth_{DBc}(s : S, sr : IDP \cup CDP) = True}$ *and has keys* $\mathrm{k} \in \mathrm{K}$ *such as* $\mathrm{OP_{DF}(ob \in OB | ob_k \in K | s_k \in K) = True}$.

3.3 Example Scenarios

In this section we will describe the operation of HMBAC through two examples, in the Supply Chain and Medical Ecosystems.

Pharmaceutical supply chains are typical examples of multi-authority and multi-domain ecosystems. In these ecosystems, various roles with often contradictory interests exist, like manufacturers aiming for rapid production while regulators enforce stringent quality controls. The HMBAC model is adept at managing the complex requirements of access permissions within such environments.

Assume, for example, that there are two authorities: the Regulatory Authority and the Customs Authority. The Regulatory Authority ensures compliance with health and safety standards, requiring access to manufacturing records and audit trails to verify adherence to laws. The Customs Authority manages import/export controls and needs access to shipping logs and batch records to ensure that prohibited substances are not transported across borders.

In the domains of Manufacturing, Distribution, and Retail:

- Manufacturing involves employees who need access to production data and control systems but should not access financial sales data.

- Distribution includes logistics providers who require access to shipping information to efficiently route products but do not need access to detailed manufacturing processes.

- Retail involves pharmacy staff who need access to inventory data and sales records but should have no access to manufacturing details.

Each user and object in the HMBAC system is assigned specific attributes:

- Users receive attributes based on their role, domain, and specific operational conditions.

- Objects like data files and systems are categorized by sensitivity and type, with corresponding access requirements.

Effective attribute derivation functions, such as effectiveUA_u for users and $\text{effectiveS}_{\text{att}_\text{u}}$ for subjects, dynamically adjust attributes based on contextual information such as time or location. The Attribute Verification Function (AVF) initially verifies that a user's attributes comply with the policies associated with the desired access. Subsequently, the Authorization Function (AF) determines if the adjusted attributes meet all the conditions set by cross-domain policies and current operational requirements. For example, a logistics provider that attempts to access shipping logs may have their attributes verified and authorized only during their operating hours under secure conditions.

Furthermore, a Decryption Function (DF) ensures that sensitive data, such as encrypted batch records, can only be accessed by authorized personnel under proper conditions. This protects against unauthorized access and ensures data integrity throughout the supply chain.

In general, the HMBAC model facilitates a flexible, yet secure framework that supports the varying needs of each stakeholder in the pharmaceutical supply chain, ensuring that access is granted appropriately based on real-time conditions, roles, and compliance requirements.

Medical ecosystems are complex networks involving multiple authorities and domains, where secure and precise access control is critical. These ecosystems consist of diverse roles, such as healthcare providers seeking to offer timely medical interventions, insurance companies assessing claims, and regulatory bodies overseeing compliance and patient privacy laws. The HMBAC (Hierarchical Model-Based Access Control) model effectively manages these intricate requirements of access permissions.

Consider, for example, three primary stakeholders: Hospital Administration, Health Insurance Companies, and Regulatory Authorities:

- Hospital Administration: that manages clinical and administrative data, requiring granular access control to protect sensitive patient information while ensuring that healthcare providers have timely access to medical records.
- Health Insurance Companies: that need access to certain patient data for claim processing and fraud prevention but must do so in a manner that respects privacy regulations.
- Regulatory Authorities, such as the Health Department or Privacy Protection Agencies, that need oversight access to ensure compliance with health care standards and privacy laws.

In this medical ecosystem, domains such as Clinical Operations, Billing Departments, and Patient Records are established:

- Clinical Operations involve doctors, nurses, and medical technicians who require real-time access to patient health data but should not access financial billing details.
- Billing Departments handle insurance claims and patient billing; staff in this domain need access to treatment details and insurance policy information but not to in-depth medical diagnostics.
- Patient Records are managed to ensure confidentiality and integrity, with access strictly controlled to prevent unauthorized viewing or tampering.

The HMBAC model in this setting would involve, (i) Attribute Assignments: Users (e.g., doctors, billing clerks) and objects (e.g., digital health records, insurance claim files) are assigned attributes based on their roles, domains, and the data's sensitivity and (ii) Effective Attribute Derivation Functions: Functions like $\mathrm{effectiveUA_u}$ dynamically adjust user attributes based on factors such as current location (e.g., hospital wards, administration offices) or time (e.g., during a medical emergency).

Initially, the AVF, checks to ensure that a user's attributes match the access policies associated with healthcare data or systems they wish to use. Then the AF checks that the user's context-adjusted attributes meet all necessary conditions imposed by inter-domain policies and operational needs. For example, a doctor in the emergency department may be granted expedited access to patient records during a critical situation. DF, guarantees that particularly sensitive data, such as psychiatric evaluations or personal health information, can be accessed only under strict conditions, thus safeguarding patient confidentiality.

CHAPTER 4

SYSTEM DESIGN

The system architecture is designed with real-world requirements in mind, notably these of supply chain and medical sector. Note, however, that other digital environments can easily be supported. The hierarchical multi-blockchain access control model discussed in the previous chapter sets the stage for this chapter, where we delve into the architectural nuances and the comprehensive design of the system. This chapter, aims to meticulously outline three critical building blocks that constitute the core of the HMBAC system. The first building block of our system is the user interaction component, called *Frontend Layer*, which is the gateway through which users interact with the system, performing operations such as submitting data access queries and receiving responses. The second, is the Blockchain Infrastructure, which is foundational to the operation of the access control system and consists of the Proxy Blockchain and several Domain Blockchains. This infrastructure not only supports the security services but also hosts the Smart Contracts that govern interactions within the system. The third and final building block of our system is the Data Layer. This layer contains the individually managed databases where the actual data resides. It is crucial for the storage, retrieval, and management of data in a secure, efficient, and scalable manner. Each of these components is designed to seamlessly integrate with others, ensuring robust security and efficient performance while facilitating user interaction and data management within a decentralized environment. In order to describe the proposed architecture in a coherent way, we shall follow the guidelines proposed in ISO/IEC/IEEE 42010:2011.

4.1 High Level Description

The goal of the proposed architecture is to develop an efficient system for managing data in MA and MD ecosystems throughout their life-cycle. In our design, we take into account the requirements of real-world environments and stakeholders,for example for the healthcare ecosystem we consider hospitals, medical device manufacturers and insurance companies, while for the supply chain logistic, transportation and retail companies. In addition, we consider various types of users or devices that may need to read or write related data.

The system is based on the novel hierarchical multi blockchain access control model presented in Chapter 3. At the user layer, an API allows all users (and 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 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). In the example of the medical ecosystem, 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. An example is shown in Fig. 4.1. In the proposed architecture, the DBCs are subordinate blockchains and cannot communicate directly with users. The PBC preprocesses 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.

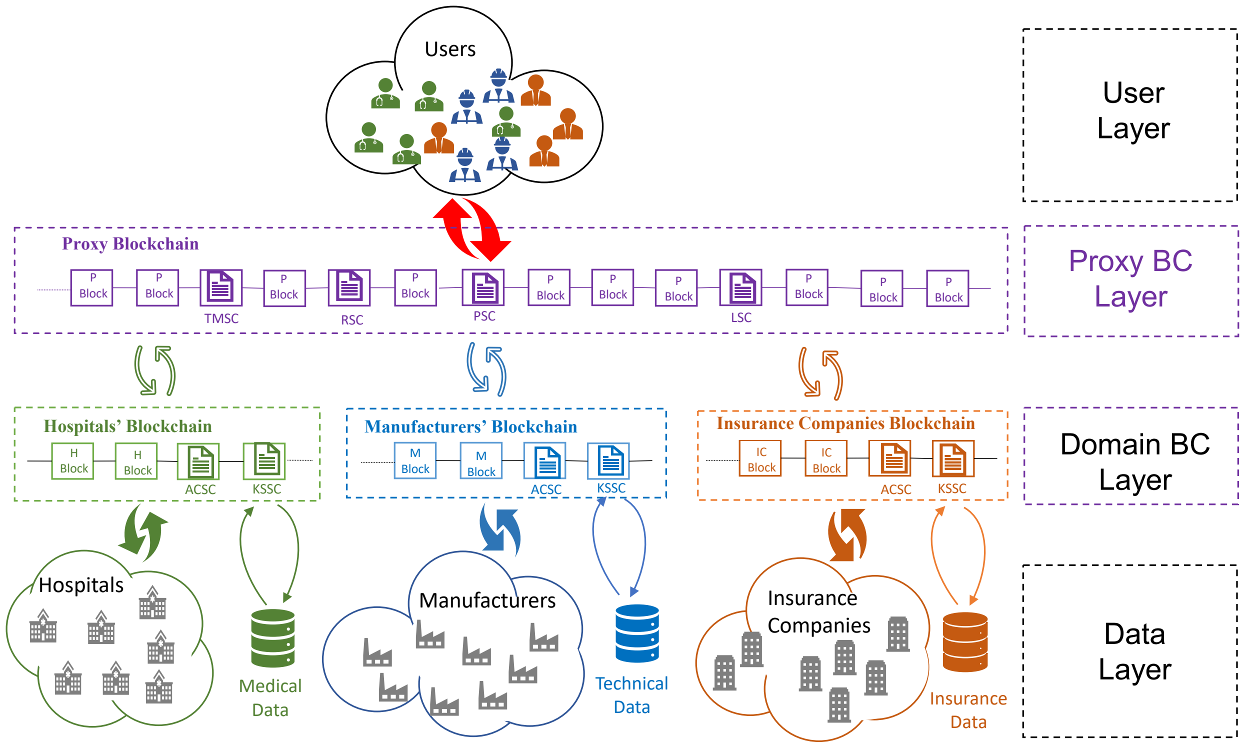


Figure 4.1: High level architecture design for the healthcare use case

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) 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.

4.2 Assumptions and Setup

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

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.

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).

ABE 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. In our system, we chose the Lewko-Waters' Multi- Authority CP-ABE (MA-CP-ABE) scheme [97], 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

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

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, an example from the medical ecosystem is shown in Fig. 4.2 .

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

Figure 4.2: A high level description of the MA-CP-ABE scheme employed in the system

As observed in Fig. 4.2, 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. 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 the 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** $\big($X.allow(α, MHR(β)), $\forall$ MHR(β) $\in \mathrm{DB_X}\big)$

2. **IF** α.isDoctorAtHospital($*$) **AND** α.onDuty($*$) **THEN** $\big($X.allow(α, MHR(β)), $\forall$ MHR(β) $\in \mathrm{DB_X}\big)$

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.

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 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.

Combining credentials with ABE keys. Fig. 4.3 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.

[2]Revoking static roles is performed at authority level, by revoking the corresponding user certificate.

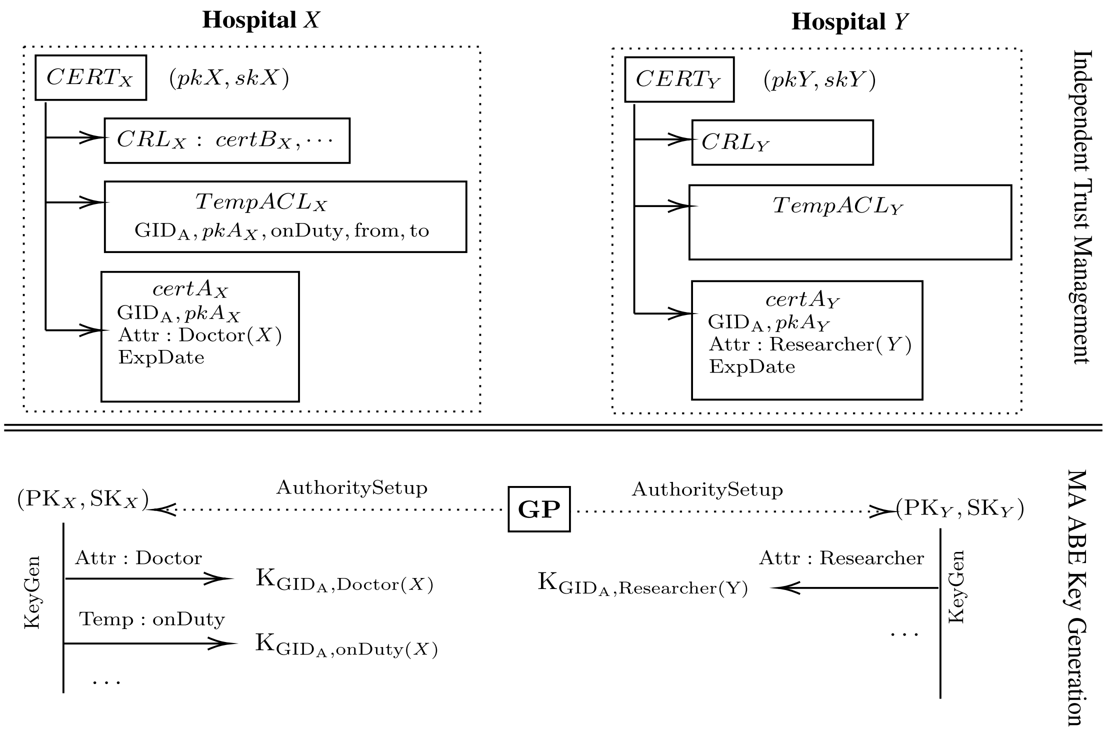


Figure 4.3: Credential issuing and ABE key generation

As shown in Fig. 4.3, 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 $\mathrm{GID_A}$ 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 $\mathrm{K}_{\mathrm{GID_A},\mathrm{Doctor}(X)}$ and $\mathrm{K}_{\mathrm{GID_A},\mathrm{Researcher}(Y)}$ 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 $\mathrm{K}_{\mathrm{GID}_\mathrm{A},\mathrm{onDuty}(X)}$ (obtainable from $TempACL_X$). Note however that the actual ABE key $\mathrm{K}_{\mathrm{GID}_\mathrm{A},\mathrm{onDuty}(X)}$ 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.

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. 4.4 describes the storage, updating and indexing of the root certificates (a similar process is used for the CRLs and temporal ACLs).

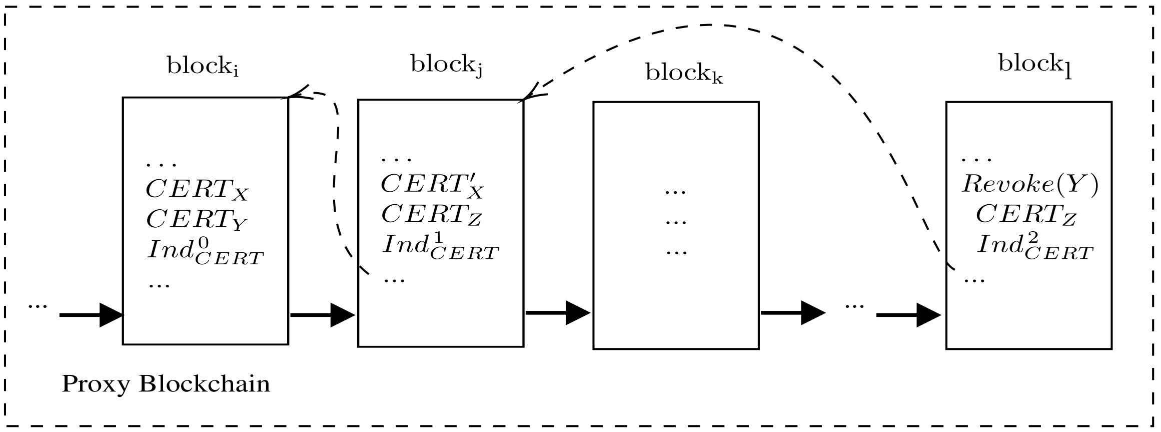


Figure 4.4: Indexing trust anchors in the Proxy Blockchain

During the initial system bootstrap, the root certificates of all stakeholders are uploaded to the PBC, say in block$_\mathrm{i}$. 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 $Ind_{TempACL}$.

4.3 System Architecture

Taking into account all the assumptions and prerequisites described previously, Figure 4.5 describes the proposed HMBAC architecture, as well as its mapping to the generic HMBAC model (presented in Figure 3.1). The architecture is comprised of three building blocks. The *Frontend Layer*, is a web application which allows authorized users to interact with the system and post data access queries. It consists of a web user interface (UI) and front-end services that support the communication between the front-end and the rest of the system.

The *Blockchain Infrastructure* is a middleware that implements all system services and provides controlled access to data, which are maintained off-chain individually by each stakeholder. It implements the hierarchical multi-blockchain, which consists of one Proxy Blockchain (PBC) and one or more Domain Blockchains (DBCs). The PBC acts as a single access point for users, while the DBCs enable the management, implementation, and enforcement of flexible and granular access policies at the domain level. The integration of the blockchain components is supported by special-purpose APIs both for inter-blockchain synchronization (Inter-BC API), user interaction (Frontend API), and data (Database API).

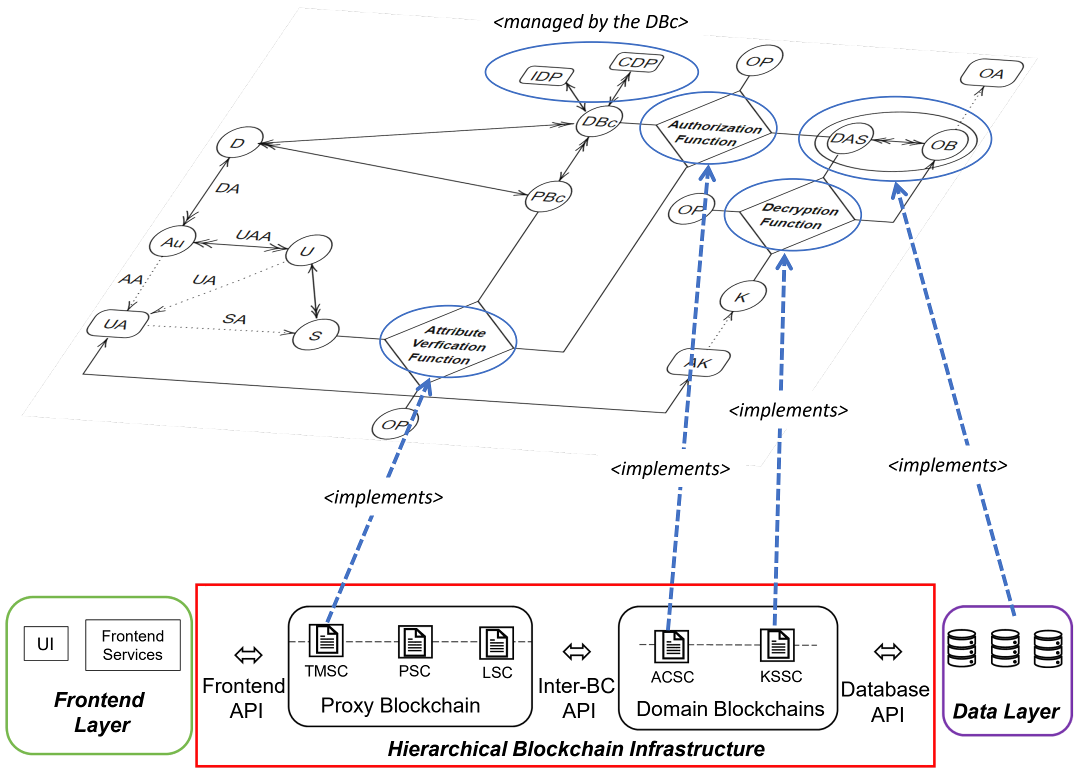


Figure 4.5: The proposed HMBAC architecture combined with the HMBAC model of Figure 3.1.

Finally, *Data Layer* contains the individually managed databases that store all data off-chain and ABE encrypted. As the data are encrypted with ABE and the decryption process requires partial decryption through the corresponding DBC, the system enforces data access through the HMBAC system. Note that CA management at the stakeholder level is managed outside our system and our main goal is to offer credential interoperability, i.e., credentials issued from independent authorities are mutually trusted, without assuming a globally trusted root authority. In the following subsection, we describe in detail the services provided by each building block.

Delving deeper in the various components, *Frontend Layer* is a web interface enables users to log in to the system and post data access queries. To access the system, two-factor authentication is enforced: the user must provide valid login credentials (e.g., a password) and a valid attribute certificate issued by a stakeholder. We assume that user credentials are managed individually and stored securely by each user. The communication between the front-end layer and the blockchain infrastructure services is realized by endpoints implemented as Frontend Services and Frontend API. Finally, when the user eventually receives the partially decrypted response data via the Frontend API, the Frontend Services will grant access to the user of the attribute keys needed to fully decrypt the data.

Data Layer is the components responsible on how the stakeholders manage their data off-chain. Recall that data are MA-ABE encrypted on the basis of predefined domain or cross-domain policies. To enforce access to data only through HMBAC, for each domain a distinct *domain attribute key pair* is assigned. During the ABE data encryption, all policies are modified by applying an additional 'AND' rule with the corresponding domain attribute key. Lets assume for example two domains *hospitals* and *medical device manufacturers.* Hospitals may involve users with various roles such as doctors, emergency doctors, or researchers. Similarly, manufacturers may support various roles, such as device technicians. According to the access control policies that may be defined within a domain or cross-domain, granular access may be allowed. For example:

- *Access Rule 1: A doctor on duty may access all medical records in all hospitals of a patient under emergency treatment* (hospital domain access rule).
- *Access Rule 2: A manufacturer's support technician may read or update the firmware of supported medical devices installed in any hospital* (cross-domain access rule).

For *Access Rule 1*, for decryption, would require the following attribute keys: K_{doctor}, K_{onDuty}, and $K_{hospitals}$. The key K_{doctor} corresponds to a long-term attribute and K_{onDuty} to a temporal attribute. The $K_{hospitals}$ is the hospital domain attribute key, generated using the hospital domain's attribute key pair PK_H, SK_H.

Finally, is the *Hierarchical Blockchain Infrastructure.* As seen in Fig. 4.5, the *Proxy Blockchain* (PBC) receives user requests through the Frontend API and implements three main services through smart contracts. User requests along with the provided attribute certificate(s) are handled by *Proxy Smart Contract* (PSC). The PSC will first trigger a certificate validation process, executed by *Trust Management Smart Contract* (TMSC). The TMSC will validate the long-term (and possible temporal) attributes assigned to users via a typical challenge-response signature verification process. Note that users can access the HMBAC services only via an *authenticated channel* (the Frontend API) and after successful *attribute authentication* (by the PBC). If user attributes are verified, the PSC will then forward the request to the relevant Domain Blockchain for further processing and wait to receive the response via *Inter-BC API*. The response will eventually be sent back to the user via the Frontend API. The transaction history is recorded on the PBC. The *Logging Smart Contract* (LSC) creates a log for each incoming user request until the transaction is completed.

Domain Blockchains (DBC) may receive user requests only from the PBC via the Inter-BC API and implement the following services. Each DBC contains an *Access Control Smart Contract* (ACSC) that enforces the access control policy of the particular domain. This includes both intra-domain and cross-domain access policies that control access to data maintained by the domain's stakeholders. The ACSC checks if the user attributes (already validated in the PBC) are sufficient for the specific request, according to the predefined access policy. In this case, the request is forwarded to the relevant database(s) via *Database API*. Note that depending on the request, the API can retrieve ABE-encrypted data from multiple sources, for example, when a user is requesting data from multiple stakeholder databases.

When the encrypted data return from the Data Layer, the Database API passes them to *Key Store Smart Contract* (KSSC), which has access to the relevant domain's attribute public/private key pair (e.g., PK_H, SK_H in the case of the Hospital DBC). The KSSC will first use the private key SK_H to generate the hospital domain attribute key ($K_{hospitals}$) based on the GID of the requesting user, to partially decrypt the data. The partially decrypted data will be forwarded to the PBC (via the Inter-BC API) and eventually to the user (via the Frontend API). The user must finally apply his attribute keys to fully decrypt and access the data (i.e., the attribute keys K_{doctor} and K_{onDuty} in our example).

4.4 Smart Contracts Design

The functionality of the system is implemented by Smart contracts that are published on the relevant blockchains (PBC, DBCs). In this section we define, from a design perspective, these smart contracts in accordance with the requirements of the system and describe their functionality per Blockchain.

4.4.1 Proxy Blockchain Smart Contracts

The PBC maintains three Smart Contracts accessible for all stakeholders of the ecosystem, namely the *Trust Management Smart Contract (TMSC)*, the *Registration Smart Contract (RSC)*, the *Proxy Smart Contract (PSC)* and *Logging Smart Contract (LSC).*

The *TMSC* handles the functionality related to the dynamic update of trust anchors (involving $CERT$s, CRLs 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$_{\mathrm{l}}$ for the root certificates). The TMSC implements the following three functions:

- *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. 4.4, the root certificate of X is updated in block$_{\mathrm{j}}$. A CA-Admin may independently update any trust anchor. Temporal access lists are expected to be updated more frequently than CRLs and root certificates.

- *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. 4.4, 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.

- *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.

The *RSC* handles the user (or the 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

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

CA authority, say Y, then A will prove possession of the global identifier $\mathrm{GID_A}$, 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 $\mathrm{GID_A}$, 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.

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.

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.

4.4.2 Domain Blockchains Smart Contracts

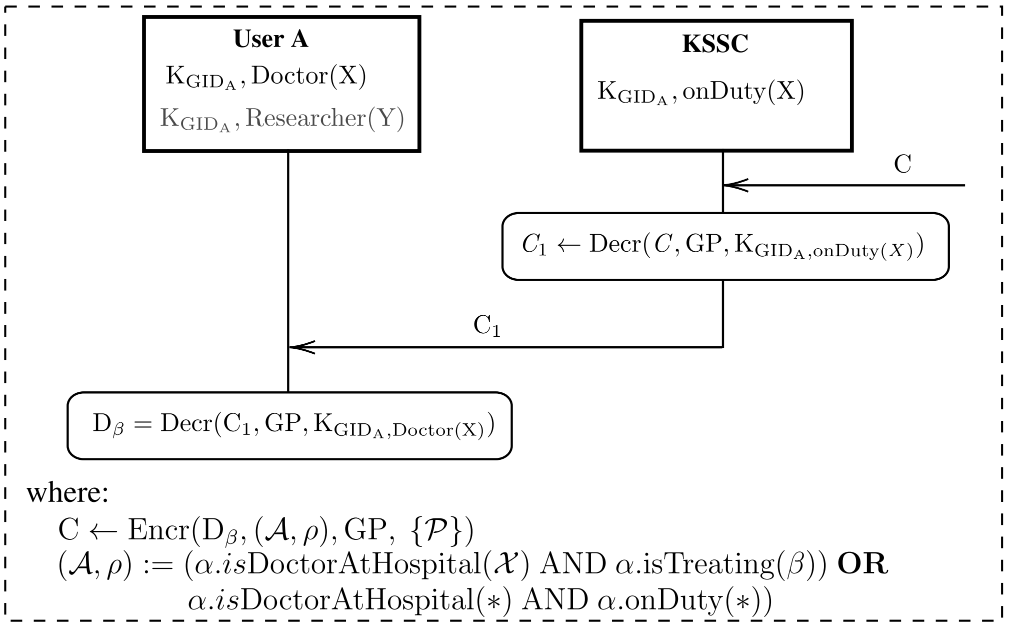
As explained in 4.3, the stakeholders of each domain (e.g. hospitals, manufacturers, insurance companies) maintain a different DBC whose function is to provide controlled and fine-grained access to its data, according to the predefined inter-domain and intra-domain policies. This structure allows different 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 DBC nodes.

Through smart contracts, the 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. Two smart contracts are implemented to provide these functions, namely, the *Access Control Smart Contract (ACSC)* and the *Key Store Smart Contract (KSSC)*.

Before granting access to encrypted data, the *ACSC* enforces the predefined access policy. 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 the Section) below 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.

As pointed out earlier, 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 these keys. Since temporal roles are always used in combination with the corresponding long-term roles , the decryption is performed as a two-step process, as shown in Fig. 4.6.



User A
K_{GID_A}, Doctor(X)
K_{GID_A}, Researcher(Y)
KSSC
K_{GID_A}, onDuty(X)
C
$C_1 \leftarrow \mathrm{Decr}(C, \mathrm{GP}, K_{GID_A,\mathrm{onDuty}(X)})$
C_1
$D_\beta = \mathrm{Decr}(C_1, \mathrm{GP}, K_{GID_A,\mathrm{Doctor(X)}})$
where:
$C \leftarrow \mathrm{Encr}(D_\beta, (\mathcal{A}, \rho), \mathrm{GP}, \{\mathcal{P}\})$
$(\mathcal{A}, \rho) := (\alpha.is\mathrm{DoctorAtHospital}(\mathcal{X})$ AND $\alpha.\mathrm{isTreating}(\beta))$ **OR**
$\alpha.is\mathrm{DoctorAtHospital}(*)$ AND $\alpha.\mathrm{onDuty}(*))$

Figure 4.6: Distributing the decryption functionality among users and the KSSC

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 $K_{GID_A,\mathrm{onDuty}(X)}$, and then will send the result to the user to finish the decryption by using the key $K_{GID_A,\mathrm{Doctor}(X)}$. The use of the MA CP-ABE [97] encryption scheme allows sequential data decryption.

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.

4.4.3 Transaction Flows

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

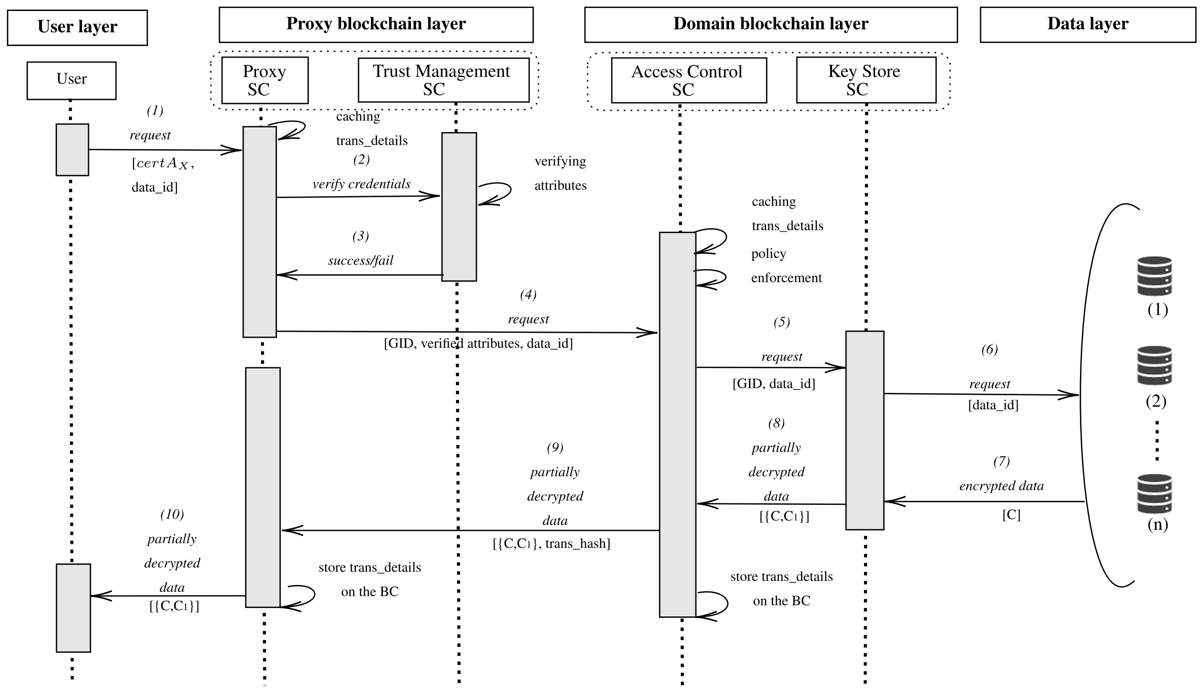


Figure 4.7: Access Control sequence diagram

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. 4.7. 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.

4.5 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 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.

CHAPTER 5

IMPLEMENTATION OF THE HIERARCHICAL MULTI BLOCKCHAIN SYSTEM

Following the architecture design presented in Chapter 4, in this chapter we analyze the implementation of *Janus* system in Malamas et al. [4] which is an integrated system for trust management and access control, suitable for MA and MD environments based on HMBAC model. The building blocks of the system are presented along with the most important aspects of the implementation design related to: isolation and orchestration between the blockchains.

The implementation of the system relies on the integration of various technologies. For implementing the Frontend component, an *Electron* [102] application was developed, while blockchains (PBC and DBCs) were developed on the Hyperledger Fabric platform, with *Raft* [103] as the underlying consensus mechanism. The functionality is implemented through Smart Contracts developed in Javascript (the full open-source implementation can be found in Malamas et al. [104] as a reproducible artifact).

Janus orchestration is based on Kubernetes [105]. For security and design modularity, all the components of the multi-blockchain infrastructure were developed in distinct Kubernetes Pods, thus providing software isolation and containerization. In particular, each smart contract, as well as the Frontend, the Inter-BC and the Database APIs are executed as separate Pods. To secure the interaction between Pods, an Ingress API supports TLS termination between the Pods.

To support the deployment of independent certificate infrastructures, each participating stakeholder establishes and maintains a *Certificate Authority* (CA), responsible for issuing and revoking certificates for their users who interact with the system. To simplify the deployment, one CA is considered to be a mutually trusted authority and is responsible for issuing certificates used by the system components (e.g., for Pods' TLS connections). Although this is not a strong requirement (e.g., it can be removed by applying cross-certification between the stakeholders), it simplifies the deployment process and it is a reasonable assumption for multi-domain environments. For example, in the healthcare sector, the ministry of health could play the role of mutually trusted authority or the Ministry of Transportation for the supply chain sector accordingly. In our system, we use *Hyperledger Fabric Certificate Authority* (provided by the Hyperledger platform) to implement the CAs of the stakeholders. Each CA runs as an instance for each entity in a separate Kubernetes Pod.

Hashicorp Vault [106] is used as an external application to issue, manage and store user and authority credentials and keys. Although it is an off-the-shelf solution, we designed an ABE plugin for Vault, written in Golang, to implement a two-step ABE decryption and support the partial decryption process via the KSSC. Additionally, we run different Vault instances, one for simulating user-side attribute key Vault storage and another for storing domain attribute keys, which is accessible only by the KSSC of each Domain blockchain.

Finally, the Frontend and the Inter-BC APIs use an instance of *RabbitMQ* [107] software, also executed in a separate pod, to temporarily store requests that remain pending in queues. The Janus implementation design is depicted in Figure 5.1 and is described in detail in the following subsections.

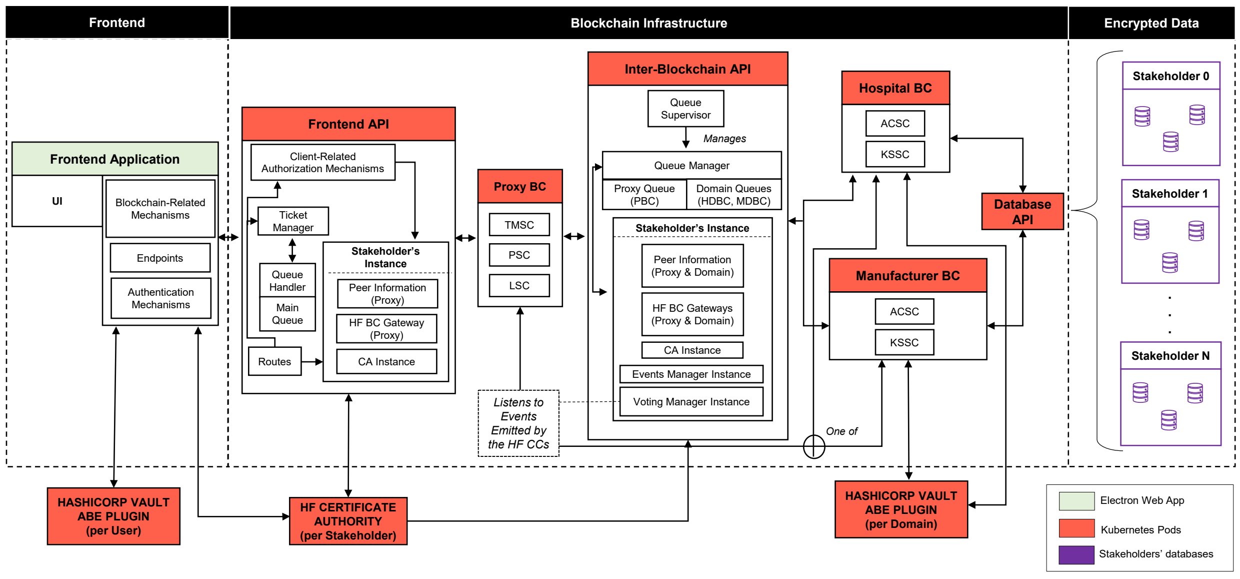


Figure 5.1: The Janus implementation design based on the HMBAC architecture.

5.1 Frontend Application

Frontend, is a client-side Electron application, running locally by each user. The *User Interface* (UI) implements the presentation layer through a web-based interface developed in React.js. enables users to log into the system using their appropriate credentials and, upon successful validation, to submit their requests. In addition, the Electron app implements various support frontend services, developed in Node.js and running in the background at the client side. The *Blockchain-Related Mechanisms*, encompass all the functionality required for a user to communicate with the middleware. These mechanisms enable a user to create and sign a query and commit it to generate a new transaction.

The *Authentication Mechanisms* supports user authentication to the system. To authenticate the credentials (attribute certificates) of a user, the Certificate Authority (CA) of the relevant stakeholder must be involved. The frontend authentication mechanisms pass the credentials along with a signed challenge to the Frontend API, which communicates with the relevant CA in order to issue a token for the user and establish communication. The authentication mechanisms also support communication with the user's Vault instance, securely storing the user-side attribute keys. Finally, *Endpoints* represent the communication points between the UI and the Frontend API. Any traffic towards the API will be handled by Kubernetes and the Ingress web server.

5.2 Frontend API

The main challenge of the Frontend component, is to implement a secure and authenticated communication channel per stakeholder, allowing users to communicate with the PBC via the Frontend application. In this way, users with valid credentials may login to the Janus services, only via the organisation's authenticated channel. To achieve this, for each stakeholder, an instance is created including information about the node (*Peer Information*), the corresponding stakeholder's gateway (*HF BC Gateway*) and the instance of the CA (*HF Certificate Authority*) where the stakeholder's certificates are stored. Furthermore, *Client-related Authorization Mechanisms*, generate the authentication token issued by the CA to which the user belongs. This token is used to establish connections between the user and the PBC.

When a user submits a request, the request is received by the *Routes* module and then queued in the *Main Queue* until served. The *Queue Handler* is responsible for the storage and retrieval of data to and from the Main Queue. When the Queue Handler authorizes the request to be forwarded, the request is retrieved from the Main Queue, and *Ticket Manager* generates a ticket for the user. Note that the Ticket Manager constantly checks for expired tickets.

The Frontend API is a RESTful API developed in Node.js using the Express framework. Since the Frontend API is running on a distinct Kubernetes Pod, its services are accessible to other Pods via the Kubernetes-exposed ports. In order to expose services on the web, an Ingress controller is used, and the communication is TLS encrypted. For increased security, the TLS session is encrypted end-to-end between the Frontend application and the Frontend API. Thus, TLS is terminated on the Frontend Pod itself instead of the Ingress controller of the API.

5.3 Inter-Blockchain API

The goal of the Inter-Blockchain API is to enable the interaction between the Proxy and the Domain Blockchains. This however raises performance and management challenges at the implementation level. Handling and prioritizing requests between different blockchains is a difficult task, as it may result in a significant performance decrease or even system failure. To deal with this issue we have implemented a novel queuing mechanism for Request Handling and Prioritization. In addition, reaching management decisions for blockchain actions that require the agreement of stakeholders at a domain or at a global level (e.g., adding/removing stakeholders or domains) requires a distributed and asynchronous decision support mechanism. To achieve this, we have implemented an efficient voting management mechanism, integrated into this layer. The implementation of these key Inter-Blockchain mechanisms is described below. Note that the Inter-Blockchain API is also developed in Node.js. Each stakeholder runs a different instance, executed on a distinct Kubernetes Pod, while its services are accessible to other Pods via the Kubernetes-exposed ports.

(a) Request Handling and Prioritization To manage requests, the Inter-BC API, utilizes two general types of queue. The *Proxy Queue*, for every request that needs to be forwarded to the PBC and *Domain Queues*, for requests that need to be forwarded to one or more DBCs. Every DBC has its own Domain Queue, and thus their number depends on the number of DBCs. The request handling flow is illustrated in Figure 5.2 and is described below.

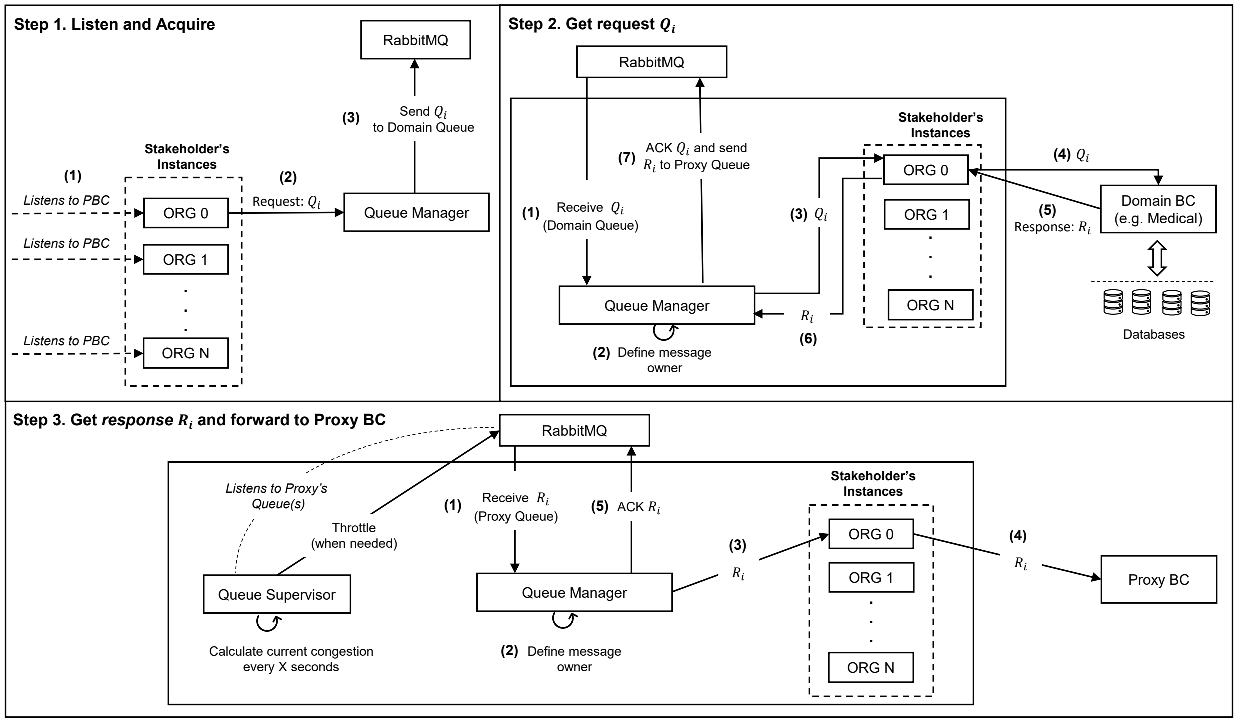


Figure 5.2: Inter Blockchain API Flow.

Step1: Listen and acquire. When an organization (*Stakeholder instance* in the Inter-Blockchain API) receives an event, it first verifies if it is the intended recipient. Next, it forwards the user query, say Q_i, to *Queue Manager* in order to temporarily save the message and process it later when it is ready for consumption.

Step2: Get the request Q_i and forward it to the relevant DBC. When it is time for a request Q_i to be consumed by the appropriate DBC, the Queue Manager receives it (from the relevant Domain Queue of the Inter-BC API) and forwards it to the instance of the organization to which the requesting user belongs. The organization instance then forwards Q_i to the DBC of which the organization is a member. Upon completing the request, the response that was received from the DBC is then, again, sent to the Queue Manager in order to queue the new message and consume it, i.e., forward the response to the PBC when ready. The Queue Manager also sends an Acknowledgment message (ACK) to RabbitMQ to inform it that the message was successfully consumed. RabbitMQ receives the ACK and removes the message from the Domain Queue.

Step3: Get the response R_i and forward it to the PBC. When the response R_i to the query Q_i is received from the relevant DBC, it is ready to be consumed. The Queue Manager receives it through *Proxy Queue* and forwards it to the appropriate organization instance. The organization then receives the response and forwards it to the PBC to continue processing the response. When forwarding R_i back to the PBC, the Queue Manager sends an ACK to RabbitMQ to inform it that the message was successfully consumed. RabbitMQ receives the ACK and removes the message from the Proxy Queue. To maintain the flow healthy and avoid Denial-of-Service errors or attacks on the Proxy BC, a *Queue Supervisor* is utilized. The Queue Supervisor constantly monitors the Proxy Queue for spikes/congestion in the network. As shown in Table 5.1, we define five types of congestion, each of which is chosen based on the current level of congestion (CL).

Table 5.1: Types of congestion.

Congestion Types			
Scale	**Congestion Level (CL)**	**Monitor Interval**	**Throttle Multiplier**
Normal	$2 \leq CL$	5 s	1
Low	$1 \leq CL < 2$	4 s	0.7
Medium	$0.5 \leq CL < 1$	3 s	0.4
High	$0.3 \leq CL < 0.5$	2 s	0.1
Extreme	$CL < 0.3$	1 s	0.01

Depending on the type of congestion, the system adjusts, in order to handle requests, avoiding DoS, and assuring that all requests will be served. The Congestion Level CL is calculated using the formula:

$$CL = \frac{max\#ConcurrentRequests}{\#QueuedRequests}$$

This means that, for example, if the PBC receives 1500 requests and accepts 400 concurrent requests, then 1100 of them will be pending in queue. The congestion level is then: $CL = 400/1100 = 0.36$, which is *High* according to Table 5.1. The system willlower *monitor interval* from 5 s to 2 s, and throttle requests sent from IBC-API to PBC will be recalculated with multiplier 0.1. If the default value is 400, then 40 concurrent requests will be sent until the PBC is discongested.

(b) Voting Management. To reach management decisions that require agreement between stakeholders, an asynchronous voting mechanism is implemented through smart contracts (described in Section 5.4 below). The Inter-Blockchain API enables the execution of this asynchronous model through the following components. As each stakeholder runs an instance of the Inter-Blockchain API, it deploys an instance of an *Event Manager* and a *Voting Manager*. The Event Manager continuously checks for new events related to active voting processes and expired elections. The Voting Manager fetches all the active elections related with the relevant stakeholder running the Inter-Blockchain instance, and it constantly awaits to receive relevant data (e.g., new votes). It also keeps a log of the submitted votes for each Election ID, and communicates with the PSC chaincode when majority is reached or time expiration occurs for a particular election.

5.4 Smart Contracts Implementation

As described in Chapter 4, the blockchain services are implemented through smart contracts. Five smart contracts utilize all system's functionality, at both the global and domain level. The PSC, TMSC, and LSC are stored at the Proxy Blockchain and shared among all stakeholders, while the ACSC and KSSC are stored at each Domain Blockchain and provide domain-specific functionality (see the relevant artifacts provided in [104] for details). Following the system design, smart contracts are able to interact with each other through the relevant APIs, as depicted in Figure 5.1. In particular, a logical connection between users and the PSC is achieved through the Frontend API while a similar connection among the Proxy BC's smart contracts and the Domain BCs is achieved through the Inter-BC API. In the following paragraphs we present the technical analysis for each smart contract along with the main services supported.

Proxy Smart Contract (PSC), integrates the functions for *user validation*, *stakeholders' voting* and *request forwarding*. For user validation, the $validateUser()$ function takes as input an array of *userCerts* provided by the user at login, and triggers the $getUserValidation()$ function stored on the TMSC, in order to validate the certificate(s) provided. Based on the outcome of the validation, the PSC returns the validated user's roles (short- and long-term). The voting mechanism is a crucial component of the system. Through this mechanism, stakeholders can reach management decisions including: a) adding/removing domains (and the relevant DBCs); b) adding/removing stakeholders to an existing DBC; c) modifying a cross-domain access policy; and d) giving access to logs for external auditors. Other processes that require broader consent can also be supported. For each stakeholder, the stakeholder administrator or a delegated external auditor may invoke the $majorityConsentInit()$ function to propose a new election. It first checks if another election with the same payload is active and then calculates the ElectionID based on the provided payload. A new Election instance is created and added to the ledger and also a ballot for each stakeholder. With $majorityClientVote()$, the administrator of each stakeholder votes in an Election by signing with the organizations' private key. The *updateElection()* function checks if the Election has been finalized based either on the predefined majority rules or the timeout set. The $requestAccess()$ function handles two processes. First, it constructs the *requestDetails*, which is sent to the LSC for logging. Then, sends a *requestForward* event to the Inter-BC API to complete the request.

Trust Management Smart Contract (TMSC), contains the functions that support trust management services. The $initLedger()$ function is responsible for handling the certificates and revocation lists of each stakeholder. It takes an *initPayload* argument and appends the corresponding data to the PBC. Additionally, it creates empty Access Control List (ACL) files for each organization. The $getUserValidation()$ function takes as input either an ACL file (to verify temporal roles assigned to users) or a certificate (to verify long-term user roles). Note that users may have obtained certificates issued by different stakeholders, provided that all certificates of a user include the same unique global identifier (GID). In addition, it allows for the efficient revocation of user access through attribute certificate revocation lists issued by the relevant authorities, instead of applying costly attribute key revocation techniques. Finally, it communicates with the LSC in order to record the transaction on the blockchain. To allow for interoperability of credentials issued by different stakeholders, all root certificates of all stakeholders are stored in the PBC. For the addition or removal of CAs, the functions $addCA()$ and $removeCA()$ are triggered accordingly. Note that both functions require agreement between current stakeholders through the voting mechanism. Only after agreement has been achieved through the voting mechanism, the function $updateTrustAnchors()$ will update the stakeholders' certificates in the PBC. The $majorityUpdate()$, invoked by the PSC, is called when an election ends (either by majority agreement or by timeout) to inform the TMSC.

Access Control Smart Contract (ACSC) main function is to enforce the predefined access policy when users request access to data stored within the domain. The Inter-BC API forwards the request and triggers the $policyEnf()$ function. Using the $data\_ID$ and the *roles* provided in the payload, it determines whether or not to grant access and forward the request to the KSSC.

Logging Smart Contract (LSC), enforces *single source of truth* in our system. For each data access request, the $requestLog()$ function is automatically triggered by the $requestAccess()$ function of the PSC. Two main processes are supported by the LSC, *registration* and *re-*

trieval of the logs. Log registration is utilized with the functions: *updateLog*(), for updating the details of uploaded stakeholders' certificates and temporal ACLs; *updateRequestLog*() for updating existing request records; and *majorityUpdate*() for updating election records. The *getUserRequestLog*() function implements log retrieval for users who want to access their request record. The *retrieveLogInit*() and *retrieveLogs*() functions are utilized for starting an access-granting Election, when an auditor requests access to the logs stored on PBC and the log retrieval accordingly.

Key Store Smart Contract (KSSC), enforces the *single point of access* property in our system in the following way. As described in Section 4.4.2, data are MA-CP-ABE encrypted, based on predefined access policies, which are modified to additionally require decryption with the domain attribute key. The KSSC is the only component that may access the domain's attribute private key. The *requestData*() function is invoked by the ACSC to initiate the process, only after the user's roles have been verified and connects to the Database API to forward the *data_ID* of requested data.

5.5 Distributed ABE Decryption Implementation

As described in Section 4.2, our goal is to cryptographically enforce a single point of entry for the system users; which means that even if a user has all the roles required for accessing some data, an extra layer of encryption will prevent access to the data outside the Janus system. The challenge, from an implementation perspective, is to create a cryptographic mechanism that will force users to access the data only through the application while ensuring fined-grained access according to predifined access policies and at the same time will remain resistant to collissions. To achieve this, we modified the implementation of the MA-CP-ABE scheme of [97], by distributing the decryption functionality between the user and the domain blockchain. We used as a basis the Python implementation of the original scheme in the *Charm* encryption library.

Since directly applying ABE encryption and decryption is not efficient, we have applied a hybrid encryption approach, where the data are symmetrically encrypted, while the symmetric keys are ABE encrypted. In each stakeholder database, each data item, say d_i, is initially encrypted with a distinct symmetric (AES) key k_i as: $c_i = AES(d_i, k_i)$. Then, each data encryption key k_i is encrypted by ABE, based on all access policies that allow access to the particular item, which are extended to include the domain attribute key of the relevant domain. For example, assume that personal information d_i of a patient should be available to the patient's family doctor or any doctor in the case of emergency treatment of the patient. In that case, the key k_i would be encrypted by ABE as follows:

$$e_1 = Enc\big(k_i, \mathbb{P}, \mathrm{GP}, \{PK_{doctor}, PK_{fDoctor}, PK_H\}\big)$$
$$e_2 = Enc\big(k_i, \mathbb{P}, \mathrm{GP}, \{PK_{doctor}, PK_{onDuty}, PK_H\}\big)$$

Each symmetrically encrypted data item c_i is sent to the DBC through the Database API, along with all ABE encryptions of k_i, in this example e_1, e_2. The KSSC has access to the domain's vault, where the hospital domain attribute key pair PK_H, SK_H is stored. Using SK_H it will generate *on-the-fly*, the hospital domain attribute key for the requesting user, i.e., $K_{U,\,hospitals} = KeyGen(\mathrm{GID_U}, \mathrm{GP}, attr{:}hospitals, SK_H)$ and use it to partially decrypt the data. The user will be able to actually decrypt the data, only if: (i) the KSSC has partially decrypted the data with the domain attribute key and (ii) the user has the relevant attribute keys for (at least) one of the above access policies, i.e., $\{K_{U,\,doctor},\ K_{U,\,fDoctor}\}$ or $\{K_{U,\,doctor},\ K_{U,\,onDuty}\}$.

We implemented the MA-ABE decryption scheme of [97] in Go as a Hashicorp Vault plug-in and integrated this into KSSC. The KSSC may trigger *sysDecrypt*(), executed in the domain's Vault instance, which generates the domain attribute key for a given GID and uses it to perform partial ABE decryption. In this way, the domain attribute key is accessible only for requests that have already been authorized by ACSC. At the same time, it is never given to users, to prevent off-system data access.

5.6 Isolation and Inter-Blockchain Communication

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* [108]. 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. 5.3.

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 system components. 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 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.

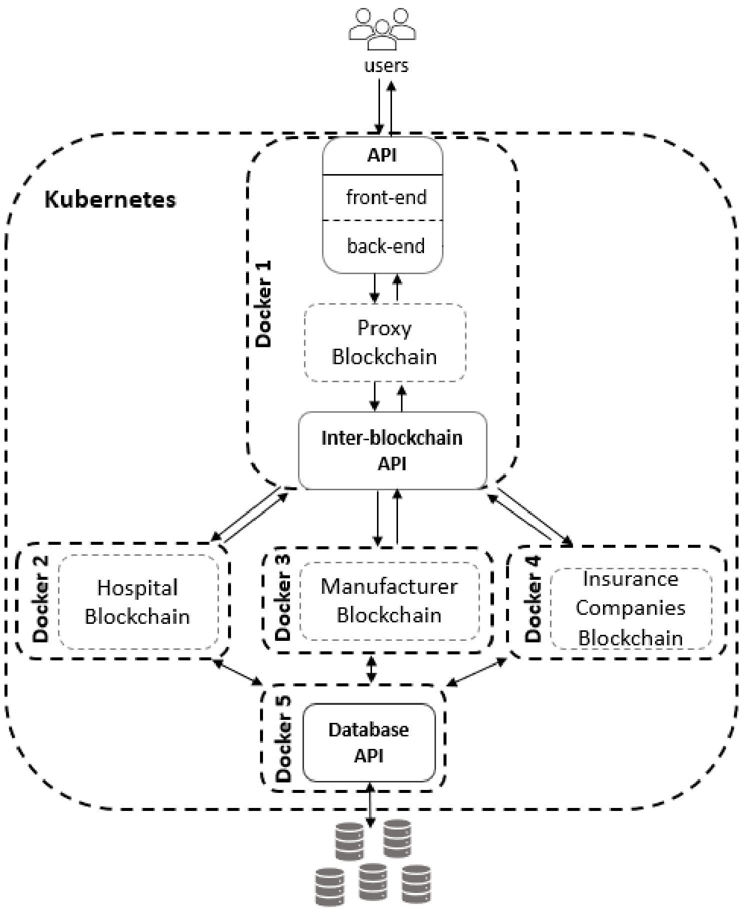


Figure 5.3: Isolation and Inter-Blockchain communication

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.

CHAPTER 6

SECURITY AND PERFORMANCE ANALYSIS

Since HMBAC is targeted to fine-grained access for multi-auhority, multi-domain environments, a practical implementation must be scalable to the number of authorities and domains. First, we analyze the scalability of the system in terms of system management. Then, we benchmark the performance of Janus for different configurations and access request rates. All measurements can be reproduced through the Janus github repository (Benchmarks are fully reproducable via an automated script – see the 'System Benchmark' section of the 'readme' document on Janus repository [104]).

6.1 Security Analysis

Threat model. We consider both internal and external attackers. Internal attackers may be compromised nodes of the HMBAC system or compromised users. Compromised nodes may attempt to illegally modify the access policies or the domain's stakeholders' set. Compromised users may attempt to bypass access control policies and gain unauthorized access. External attackers may attempt to gain unauthorized access to the system.

Assumptions. We shall assume in our analysis that the underlying software components such as the orchestration engine (Kubernetes) and the isolation mechanisms (Pods and Hashicorp Vault) are trusted. Instead of requiring a fully trusted authority, we relax our trust assumptions to a majority of trusted stakeholders for each domain. We assume that the majority of the participants in the consensus and voting protocols behave in a trusted way. We assume that the encryption and authentication mechanisms used (AES and MA-ABE) are secure (cannot be compromised by a probabilistic polynomial-time Turing machine). Finally, we assume that the user credentials are securely managed on the user side. As the main goal of HMBAC is to provide access control, we will first examine security against unauthorized access attacks and then other security characteristics of the proposed system.

6.1.1 Secure Data Access

The security of HMBAC-controlled data access is based on several security building blocks (as detailed in Section 4.3). First, data are encrypted with ABE with the keys assigned to users based on their roles, by applying the MA-ABE scheme in [97]. Then, an additional layer of ABE encryption is performed, with an attribute key assigned to the Key Store Smart Contract (KSSC). This is implemented by applying an additional 'AND' rule on top of the predefined encryption policy. This forces all requests to be performed via the HMBAC system; otherwise, the data retrieved by users will still be partially encrypted. The BC-side attribute keys are securely stored in a Vault and are accessible only by the KSSC.

Besides the encryption layer, the user must be authenticated by the system to send queries, and also by the user-side Vault to access the attribute keys, to decrypt the received partially encrypted data. System authentication is performed through the proxy blockchain using the Trust Management Smart Contract (TMSC). An authenticated user may then send a data access request, which in turn will be validated at the domain blockchain layer, via the Access Control Smart Contract (ACSC), in order to verify that the user has the required roles based on the access policy. The KMSC performs the required partial decryption. Finally, users need access to their attribute certificates, issued by the relevant stakeholders/authorities, to verify their roles with the ACSC (Note that the attribute certificates may also be stored in a user-side Vault for protection).

To formalize our analysis of unauthorized access attacks, we use attack trees as in [109]. Attack trees [110] is a conceptual design used to describe attacks on system assets. We distinguish two types of attack nodes, *and-nodes* and *or-nodes*: the children of an and-node should all be executed to reach the goal of their parent, while any one of the children of an or-node needs to be executed to reach the goal of its parent. An attack on the system is then modeled by a multi-set of compromised nodes.

Definition 4. *Let $\mathbb{C}$ be a set of attack components of a system. An attack is a finite non-empty multi-set of $\mathbb{C}$ and an attack suite is a finite set of attacks. Denote the universe of attacks by $\mathbb{A} = \mathcal{M}^+(\mathbb{C})$ and the universe of attack suites by $\mathbb{S} = \mathcal{P}(\mathbb{A})$.*

The attack tree for unauthorized data access attacks in HMBAC is shown in Figure 6.1. Our goal is to analyze all possible attack paths for an adversary, external and/or internal, to compromise the access control mechanism and gain unauthorized data access. As defined in our threat model, accessing the data in ways that are outside the HMBAC system is out of scope, e.g., accessing the data before they are ABE encrypted or before their entry into the system.

To construct the attack tree, first, we observe that unauthorized data access requires an adversary to concurrently bypass the security mechanisms that: validate a data access query posted to the PBC (denoted by node A), *and* access all the attribute keys used to encrypt the data (denoted by node B). Note that despite the actual attack that may be applied to achieve the above conditions, simultaneously achieving the attack components A and B are necessary and sufficient conditions for any successful attack on unauthorized data access against an HMBAC system. Then for each level-1 node, we continue our analysis of identifying all possible sets of system components that must be successfully attacked to achieve each goal of the relevant parent node. The same holds for all nodes of the attack trees, including the leaf nodes.

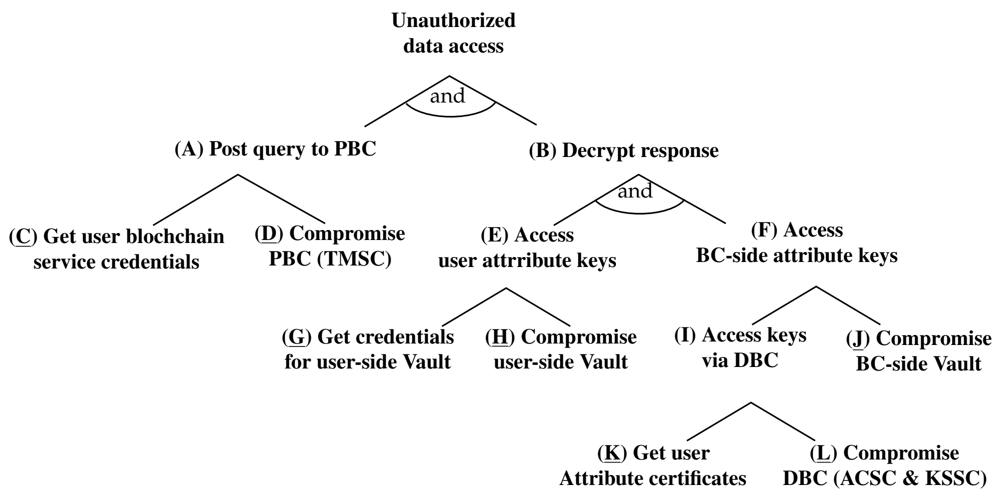

Unauthorized data access
and
(A) Post query to PBC
(B) Decrypt response
and
(C) Get user blochchain service credentials
(D) Compromise PBC (TMSC)
(E) Access user attrribute keys
(F) Access BC-side attribute keys
(G) Get credentials for user-side Vault
(H) Compromise user-side Vault
(I) Access keys via DBC
(J) Compromise BC-side Vault
(K) Get user Attribute certificates
(L) Compromise DBC (ACSC & KSSC)

Figure 6.1: Unauthorized data access tree for the HMBAC architecture

Note that for all nodes, including leaf nodes, we did not examine the actual attack techniques that may be used to achieve the relevant goal. For example, for node C there exist various implementations of attacks to obtain user credentials for the HMBC service, such as phishing, spoofing, or brute force. The goal of the attack tree analysis is to exhaustively list all possible sets of necessary attack steps (i.e., concurrently compromised security components) to succeed in the attack.

For this tree, the set of identified attack components (nodes) is:

$\mathbb{C} = \{A, B, \underline{C}, \underline{D}, E, F, \underline{G}, \underline{H}, I, \underline{J}, \underline{K}, \underline{L}\}$, with seven leaf nodes (for clarity, the leaf nodes are underlined).

Leaf nodes are vulnerable components that an attacker can exploit to initiate an attack. Any attack suite must contain such nodes, as well as the target node T.

We examine the attack suites of the unauthorized data access attack tree of the HMBAC, with respect to the successful attack steps required by an adversary. We consider the following cases:

Case 1. *Fully compromised user*: all user credentials (BC credentials or PBC access (C or D), user-side Vault credentials (G or H) and attribute certificates (K)) are compromised.

We get the attacks: $\{\underline{C}, A\}$, $\{\underline{D}, A\}$, $\{\underline{G}, E, B\}$, $\{\underline{H}, E, B\}$ and $\{\underline{K}, I, F, B\}$, that when combined give us the attack suites:

$$\begin{aligned}
S_{1cgk} &= \{\underline{C}, A, \underline{G}, E, \underline{K}, I, F, B, T\},\\
S_{1dgk} &= \{\underline{D}, A, \underline{G}, E, \underline{K}, I, F, B, T\},\\
S_{1chk} &= \{\underline{C}, A, \underline{H}, E, \underline{K}, I, F, B, T\},\\
S_{1dhk} &= \{\underline{D}, A, \underline{H}, E, \underline{K}, I, F, B, T\}.
\end{aligned}$$

The attacker will then be able to post to the system all queries available to the target user. However, this attack does not leak data from other users.

Case 2. *Partially compromised user*: at least one of the required user credentials C, D, G, H and K is secure. In this case, from the attacks: $\{\underline{C}, A, \underline{G}, E\}$, $\{\underline{D}, A, \underline{G}, E\}$, $\{\underline{C}, A, \underline{H}, E\}$, $\{\underline{D}, A, \underline{H}, E\}$, and $\{\underline{K}, I, F\}$, $\{\underline{L}, I, F\}$, $\{\underline{J},$
$F\}$, we obtain the attack suites:

$$\begin{aligned}
S_{2cgk} &= \{\underline{C}, A, \underline{G}, E, \underline{K}, I, F, B, T\},\\
S_{2cgl} &= \{\underline{C}, A, \underline{G}, E, \underline{L}, I, F, B, T\},\\
S_{2cgj} &= \{\underline{C}, A, \underline{G}, E, \underline{J}, F, B, T\},\\
S_{2dgk} &= \{\underline{D}, A, \underline{G}, E, \underline{K}, I, F, B, T\},\\
S_{2dgl} &= \{\underline{D}, A, \underline{G}, E, \underline{L}, I, F, B, T\},\\
S_{2dgj} &= \{\underline{D}, A, \underline{G}, E, \underline{J}, F, B, T\},\\
S_{2chk} &= \{\underline{C}, A, \underline{H}, E, \underline{K}, I, F, B, T\},\\
S_{2chl} &= \{\underline{C}, A, \underline{H}, E, \underline{L}, I, F, B, T\},\\
S_{2chj} &= \{\underline{C}, A, \underline{H}, E, \underline{J}, F, B, T\},\\
S_{2dhk} &= \{\underline{D}, A, \underline{H}, E, \underline{K}, I, F, B, T\}.\\
S_{2dhl} &= \{\underline{D}, A, \underline{H}, E, \underline{L}, I, F, B, T\},\\
S_{2dhj} &= \{\underline{D}, A, \underline{H}, E, \underline{J}, F, B, T\}.
\end{aligned}$$

Again, these attacks only affect the data of the compromised users.

Case 3. *Fully compromised PBC (D) and DBC (L).* Here, unauthorized queries are posted due to a compromised Proxy BC (bypassing the TMSC), while access to the BC-side keys assumes a compromised domain BC (bypassing the ACSC control and utilizing the BC-side attribute keys via the KSSC). However, a successful attack suite requires additional access to the user attribute keys, either by compromising the user-side Vault (H) or by getting the user credentials (G). We obtain the attack suites:

$$\begin{aligned} S_{3dhl} &= \{\underline{D}, A, \underline{H}, E, \underline{L}, I, F, B, T\}, \\ S_{3dgl} &= \{\underline{D}, A, \underline{G}, E, \underline{L}, I, F, B, T\}. \end{aligned}$$

Case 4. *Fully compromised Vault.* Here both the user- side and BC-side Vaults (H and J) are compromised. Again, a successful attack requires additionally a partially compromised user (C) or Proxy BC (D). We obtain the attack suites:

$$\begin{aligned} S_{4chj} &= \{\underline{C}, A, \underline{H}, E, \underline{J}, F, B, T\}, \\ S_{4dhj} &= \{\underline{D}, A, \underline{H}, E, \underline{J}, F, B, T\}. \end{aligned}$$

Case 5. *All entities partially compromised.* Here the user credentials/certificates (C, K), blockchains (D, L) and vault storage (G, H, J) are all partially compromised. We get the attack suites:

$$\begin{aligned} S_{5chl} &= \{\underline{C}, A, \underline{H}, E, \underline{L}, I, F, B, T\}, \\ S_{5dgj} &= \{\underline{D}, A, \underline{G}, E, \underline{J}, F, B, T\}, \\ S_{5dhk} &= \{\underline{D}, A, \underline{H}, E, \underline{K}, I, F, B, T\}. \end{aligned}$$

We now have:

Proposition 1. *Compromised user credentials (either fully or partially) cannot affect the data access of other users.*

Proof. This follows directly from Cases 1 and 2. □

Proposition 2. *The system can resist unauthorized data access even if both the proxy and the domain blockchains are compromised, provided that the user attribute keys are secure.*

Proof. This follows directly from Case 3. □

Proposition 3. *The system can resist unauthorized data access if at least one of the system entities (users, blockchains, key Vaults) is secure.*

Proof. This follows directly from Cases 4 and 5. □

6.1.2 Secure Blockchain Management

The security of critical management decisions that could compromise the system's security relies on (i) the voting mechanism implemented on the Proxy blockchain, (ii) the blockchain consensus mechanism, (iii) the transaction replication implemented by all the blockchains, and (iv) the execution isolation supported by the use of Kubernetes and independently managed Pods. As explained in Section 6.2 the voting mechanism, implemented by the PSC, enables stakeholders to make management decisions. Any stakeholder may start an election. Voters' eligibility and vote integrity are ensured since the private key of a stakeholder is required to sign a vote for an election. Different thresholds and eligible voters can be defined for different elections.

The blockchain consensus mechanism is also related to secure system management. Since smart contracts in both blockchain layers implement critical functionality of the system, modifying those smart contracts either at the PBC or the DBCs could compromise the security of policy enforcement. However, since smart contracts are implemented in the initial blocks of each blockchain, their integrity is strongly protected.

Since the underlying consensus mechanism of Fabric (Raft) does not support Byzantine tolerance, a malicious leader might attempt to forge the blockchain(s) logic by adding modified smart contracts, e.g., to compromise the access policy. However, such an attack would be easily detected by the other stakeholders because of the blockchain replication mechanism and the lack of integrity (valid signatures by the stakeholders' majority) of the modified smart contracts. Finally, the encapsulation of all the distributed components in replicated independent Pods, executed by different stakeholders and orchestrated by Kubernetes, also protects system integrity.

6.1.3 Secure Key Storage/Management

The use of Hashicorp Vault provides secure key storage. For each DBC, an independent vault instance is used to store and securely access the domain's attribute key. In addition, users may also deploy vault instances to protect their attribute keys and attribute certificates. Finally, certificate management at the stakeholder level is implemented by independent instances of Hyperledger Fabric CA running on different Pods. These are accessible by the TMSC through encrypted and authenticated Kubernetes ports.

6.2 System Scalability and Management

The modular design of the HMBAC architecture allows for scalable and efficient system management. Adding or removing users in Janus is handled independently by each organization (stakeholder). Each organization is able to issue attribute certificates and give access to the corresponding attribute keys to allow its users to: (i) post queries that will be accepted by the ACSC, based on the user's roles; and (ii) fully decrypt a response that has been partially decrypted by the KSSC. Adding/removing stakeholders within an organization, or changing the access policy of the domain, is handled at the domain level. Due to the use of independent DBCs per domain, managing functions within a domain will not cascade to affect the other domains. The use of the voting mechanism enables setting up elections at a domain level and in addition to define a majority threshold at the domain level for decisions affecting a particular DBC. Finally, adding new DBCs will require a majority vote by all stakeholders and will affect all domains, as this will require updating the smart contracts in the PBC.

6.3 Benchmarks

We conducted our evaluation on two different hardware configurations with varying resources, using the Linode cloud infrastructure. As depicted in Table 6.1, in the first H/W setup (S1), an AMD EPYC 7501 32-core processor @2GHz with 64 GB RAM is used. The second H/W setup (S2) is an environment with higher resources, based on an AMD EPYC 7702 64-core processor running at @2GHz with 512 GB RAM. As our implementation Janus utilizes eight (8) Kubernetes pods, where each Pod corresponds to an independently managed server, setup S1 (resp. S2) corresponds to four cores/8GB RAM (resp. 8 cores/64GB RAM) per server.

Table 6.1: H/W specs for testing.

	CPU (# Cores)		**RAM (GB)**	
	Total	**Per Pod**	**Total**	**Per Pod**
Setup S1	32	4	64	8
Setup S2	64	8	512	64

Both sets of configuration run Ubuntu 20.04.1 LTS OS and Kubernetes 1.20.11 was used for container orchestration. The multi blockchain components were developed in Hyperledger Fabric 2.4 beta with Raft as the underlying consensus algorithm and also fabric-ca-client 2.2.6, fabric-network 2.2.9 and fabric-gateway 0.1.0 were used for establishing communications.

Following the two access rule examples for the medical ecosystem, mentioned in Section 4.3, we created both inter-domain queries (e.g., *"Retrieve the medical record for patient P from all hospital databases"* and cross-domain queries (e.g., *"Update the firmware for medical device D of manufacturer M at all hospitals"*). Each database was running on a separate Pod and data were ABE encrypted. The initial ABE decryption was performed by the KSSC running in the relevant BC domain of the requesting user, as described in Section 5.5.

We measured the average end-to-end query response time for various sizes of queries, ranging from 2 up to 300 *concurrent* queries (req/sec), with an approximately even portion of inter-domain and cross-domain queries. Fig. 6.2 shows the average execution time for all scenarios tested. In addition, the table presents the time needed for the main subprocesses of the query–response process.

# of Concur. Requests	**2**		**10**		**20**		**40**		**60**		**80**		**100**		**200**		**300**	
H/W Setups	**S1**	**S2**	**S1**	**S2**	**S1**	**S2**	**S1**	**S2**	**S1**	**S2**	**S1**	**S2**	**S1**	**S2**	**S1**	**S2**	**S1**	**S2**
Ticketing	0.004	0.002	0.003	0.002	0.002	0.002	0.002	0.003	0.003	0.005	0.025	0.008	0.018	0.015	0.015	0.024	0.032	0.02
Endorse	0.08	0.07	0.25	0.12	0.24	0.17	0.16	0.23	0.22	0.15	0.48	0.55	0.44	0.18	0.55	0.3	0.82	1.44
Commit	0.006	0.007	0.006	0.004	0.005	0.005	0.006	0.005	0.005	0.005	0.008	0.008	0.008	0.025	0.010	0.010	0.014	0.020
BC_RTT	2.19	2.2	2.41	2.22	2.47	2.07	2.95	2.49	3.19	2.76	3.33	2.74	4.22	3.76	6	4.63	8.09	5.62
Average	2.27	2.27	2.67	2.35	2.72	2.25	3.12	2.73	3.43	2.92	3.85	3.32	4.69	3.98	6.58	4.97	8.96	7.12
Min	2.26	2.27	2.58	2.28	1.88	2.1	2.02	2.92	1.62	1.61	1.88	1.98	2.04	2.45	1.74	1.54	1.65	2.33
Max	2.29	2.27	2.72	2.39	2.91	2.58	3.88	4.05	4.86	3.66	5.14	5.31	5.71	5.22	9.2	7.84	12.77	10.39

Figure 6.2: Deatailed performance evaluation for variouys scenarios of concurrent requests and h/w setups (time in s)

Ticketing, refers to the time required by the system to issue a ticket for a user. *Endorse*, is the time it takes for peers to receive a request and sign the result. *Commit*, is the time required by the orderer nodes to create a new block. Finally, *BC_RTT* is the time needed to execute all the required BC functions (smart contracts) and inter-BC communication. In addition, the minimum and maximum time required for a query is presented in each scenario, to exhibit the deviation from the average time. As expected, the most resource-intensive process is BC_RTT, which encompasses all subsystems, from Proxy BC up to the retrieval of encrypted data from the independently managed databases, as well as the partial decryption process using the domain keys.

However, the overall time increase is linear (see Figure 6.3), which indicates the scalability of the HMBAC design.

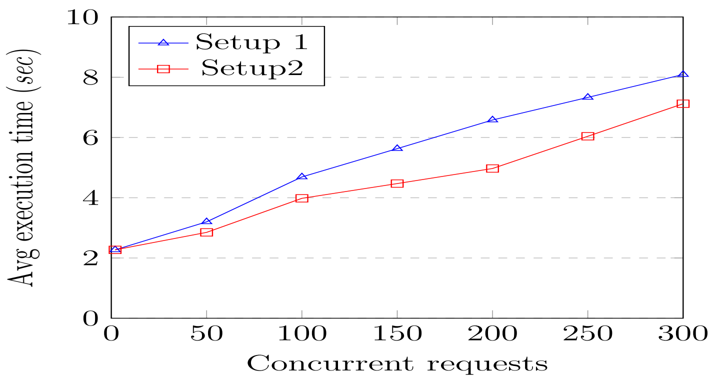


Figure 6.3: System Efficiency.

Adding new authorities will increase the number of users and, consequently, the number of requests. At the same time, it will also increase the overall system resources, as the new authorities will devote resources to become stakeholders of the Proxy and their Domain blockchain. The system's performance is linearly dependent on the available resources, which means that the overall time decreases as resources increase. Note that in both system setups, the system presents zero errors per requests, due to the queuing module integration.

SECTION III

Other security features

CHAPTER 7

IMPLEMENTING DISTRIBUTED SELF-SOVEREIGN IDENTITY MANAGEMENT

The implementation of Self-Sovereign Identity (SSI) within the Janus multi-blockchain platform represents a pivotal advancement in managing the identities of users and devices. This SSI component is meticulously designed to ensure robust, secure, and user-centric identity management. It supports selective disclosure, allowing identity holders to share only the minimum necessary information, thereby enhancing privacy and control. Additionally, the component integrates Zero-Knowledge Proof (ZKP) capabilities, enabling the verification of certain attributes, such as age, without disclosing the underlying information, such as the date of birth.

Beyond the basic functions of issuing and presenting credentials, the SSI component in Janus extends its utility to proactive identity management. It includes features such as notifying holders about revoked credentials and facilitating secure message exchanges. These capabilities ensure that the Janus platform not only provides a secure and efficient identity management system but also fosters enhanced interoperability and user engagement in a multi-authority and multi-domain environment.

7.1 SSI Component Architecture Design

In the proposed architecture we consider that the process of authentication starts when a user or a authority gets a new device and ends when the device can be treated as an authorized device and can securely exchange data with the system. For this purpose, a unique public/private key pair is certified for each device, to secure communications between devices, the IoT gateway or the central system.

High level description of the proposed architecture. The proposed architecture gives the identity holder the ability to use their digital wallet and authenticate their identity using their credentials. The main concept behind decentralized identity applications is the Verifiable Credential (VC) model, which creates trust in these applications. The most important feature of verifiable credentials and the one that makes this technology suitable for our implementation, i.e., the authentication of IoT devices, is that credentials and their presentations are not simple plain text documents. They are cryptographically built to contain all the credentials' basic key attributes (who issued the credential, if the credential data hasn't been changed, etc.). The presented credentials' verification uses data written to a distributed immutable ledger (blockchain), making forgery practically impossible.

The system consists of six types of entities. Distributed Ledgers, which are essentially distributed databases that contain all transactions in the network. Gateways (GW), which are middle devices between end devices and authority (e.g. smartphones, tablets, IoT). Manufacturers of the devices that are used as Gateways in the network called Gateway Vendors, IoT Devices, end devices that record and transmit data, Device vendors describing the manufacturers of these devices. Authorities that participate in the network and collect data from the devices.

Authentication process: Before a device authenticates to the system there are certain steps that need to take place, as seen in Figure 7.1. The GW's and Device's vendors issue credentials to the GW and Device accordingly, register their DIDs and publish the needed Credential Schemas (CS) to the ledger. The GW publishes a CS to the ledger for later use in the credential issuance procedure of the Devices and the Authorities. At this stage the device can authenticate to the system.

We present an example of the proposed architecture for the medical sector, where the devices are Internet of Medical Things (IoMT) and the authority/stakeholder is a hospital.

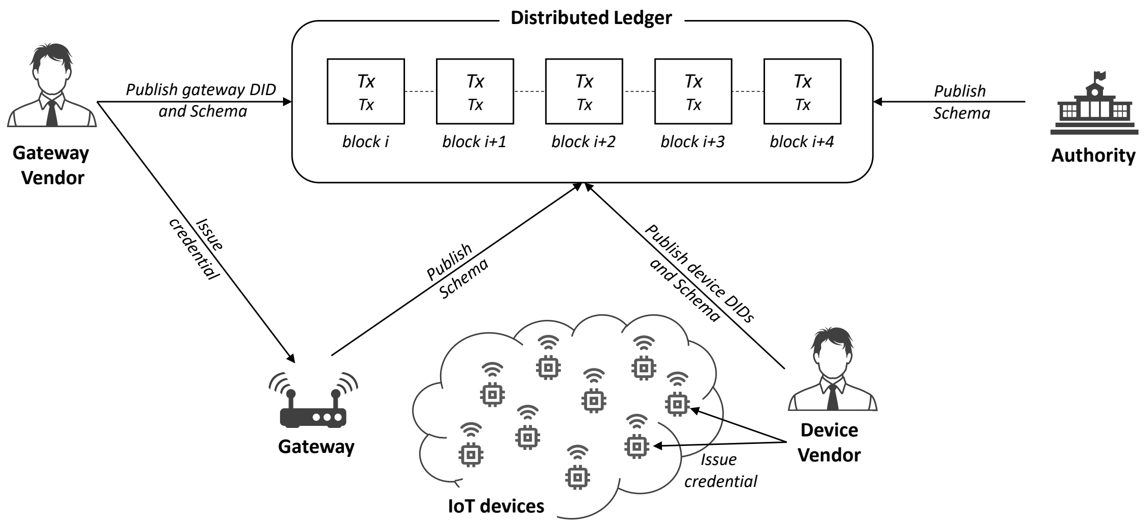


Figure 7.1: High level architecture

The full process, is depicted schematically in Figure 7.2 and is completed in three phases following the numbering. In the demonstrated scenario, Hospital initiates a connection with the corresponding Gateway using the registered Decentralized Identifier (DID) on the ledger. Once a connection is established, Hospital sends a proof-from-vendor request. Gateway sends the credential and if proof is verified, Hospital issues a credential using the CS previously published to the ledger. Gateway is now considered trusted.

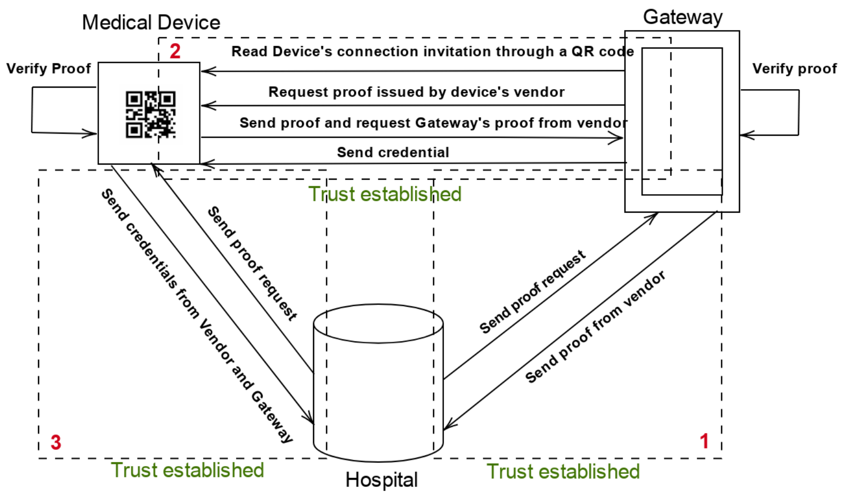

Medical Device
Gateway
2
Read Device's connection invitation through a QR code
Verify Proof
Request proof issued by device's vendor
Verify proof
Send proof and request Gateway's proof from vendor
Send credential
Trust established
Send credentials from Vendor and Gateway
Send proof request
Send proof request
Send proof from vendor
3
1
Trust established
Hospital
Trust established

Figure 7.2: Device authentication process

Then it is Gateway's turn to initiate a connection with the Device that we want to authenticate (scan an embedded QR code, input device's invitation data manually etc.). Once a connection is established, Gateway requests proof-from-vendor from Device. If the proof is verified, it then issues an authentication credential using the previously published CS. Gateway also sends Device invitation data to the Hospital so that it can initiate a connection with the Device.

In this stage, the Device owns credentials from both its vendor and a trusted Gateway. Hospital initiates a connection with the Device and requests proof-from-vendor and proof-from-gateway. Device sends the credentials and Hospital verifies them. If proofs are valid, Hospital issues an authentication credential to the Device. Authentication is completed.

Authorization after authentication: When a device is authenticated, it owns a credential issued by the hospital or a central system. Assuming there is a central blockchain or a Decentralized Ledger Technology (DLT) infrastructure to which the devices send data and, therefore authenticate, a medical device will have to initiate a connection with this system. Every time a device asks for a connection, the central blockchain will have to grant or deny access by verifying a proof that only legitimate devices will be able to construct by owning a credential from the hospital.

Revocation: As mentioned before, our approach is based on credentials. Credentials are presented and verified so that an entity can authenticate to another entity. There is also the case that an entity should not have access to the system or be authorized under a certain gateway or hospital. This can be scheduled to happen automatically after a certain period of time or it may be triggered by an event (e.g. a device changes owner, a device located in a clinic changes ward and therefore must be authenticated through a different gateway). Entities that issue credentials (Device's vendor, the Gateway's vendor, the Gateway or the Hospital (issuers)) publish revocation registries along with the credentials they issue.

7.2 SSI Implementation Details

For the implementation needs of the proposed model, we used Linux Foundation's *Hyperledger Identity Stack* (HIS), which consists of a basic framework, *Hyperledger Indy* and two tools, *Hyperledger Aries* and *Ursa*. HIS is a project that puts the concepts of SSI into practice and is quite suitable for the use-case scenario examined. We should note that, to date, Hyperledger Aries functions as an independent system that operates alongside the primary application, integrating Self-Sovereign Identity (SSI) capabilities. This configuration increases management and operational costs and complicates interoperability. In this thesis, we modified the Aries Verifiable Data Registry (VDR) to enable interaction with Hyperledger Fabric as a ledger, facilitating the development of SSI applications on the main Hyperledger Fabric platform, Janus.

Our implementation was based on the Hyperledger Aries Cloud Agent Python (ACA-Py)[1] foundation. More specifically, the Faber/Alice demo was modified and extended in a way that it meets the specifications and functioning of the architecture described in the previous section. The Verifiable Organizations Network[2] is also used as a public distributed ledger running locally for the sake of testing, its code being available on github[3].

A script called *run_demo* is executed with the proper argument (device, devicevendor, gateway, gatewayvendor, hospital) in order to initiate an agent for the corresponding entity (Medical Device, Device vendor, Gateway, Gateway vendor, Hospital). The script also creates Docker images for each entity and assigns ports to the agents. There is a controller for each entity written in Python, all of which are using methods from the agent.py controller provided by ACA-Py. In this implementation, the gateway is operated from a terminal, although in the high-level architecture it is described as a mobile device (smartphone, tablet etc.). This is to avoid any further technical complexity and because of limitations in the mobile framework of Aries at the time.

[1]https://github.com/hyperledger/aries-cloudagent-python
[2]https://vonx.io/
[3]https://github.com/bcgov/von-network

CHAPTER 8

INTERCONNECTION WITH LEGACY SYSTEMS

Companies typically view Enterprise Resource Planning (ERP) systems as a crucial component of their business operations. Inventory management, order fulfillment, transportation management and forecasting are just some of the many logistics processes that can be managed and streamlined with the help of an ERP system [111]. With the help of ERP systems, businesses can monitor their Supply Chains (SC), allowing them to respond to fluctuations in demand and supply. Furthermore, ERP systems can integrate with warehouse management or transportation systems to facilitate communication and coordination throughout the SC at every stage. Arguably, ERP systems provide several benefits to all SC stakeholders such as cost reduction, increased efficiency, better customer service and enhanced quality control [112, 113]. ERP systems are especially useful in today's globalized SC context, where companies must deal with the complexities of managing vast, multi-entity SC networks and relevant information flows.

Blockchain technology and applications within the SC management domain have attracted considerable attention in recent years [114, 115]. In particular, blockchain technology offers increased transparency and traceability since it enables the creation of an immutable record of transactions, thus making it easier to track the movement of goods throughout the SC. Blockchain technology uses cryptographic techniques to ensure that data cannot be altered or tampered with, providing a high-security level for sensitive SC information. By eliminating the need for intermediaries and enabling real-time data sharing, blockchain can reduce the time and effort required to exchange information within the SC. Besides, the usage of smart contracts can automate numerous operations resulting in additional benefits in terms of efficiency and cost reduction. Most importantly, blockchain can be integrated with ERP systems to provide a single source of truth for SC data, therefore, enabling information sharing and more efficient operations. Other benefits from integrating blockchain and ERP systems include improved data accuracy and reliability, enhanced security, increased efficiency and traceability, and enhanced collaboration among multiple SC stakeholders [116].

8.1 Introduction

Despite their significant benefits, ERP systems present several limitations such as their inability to provide extended enterprise functionality across organizational boundaries, their difficulty in adapting to the evolving demands of the SC, their inability to perform any tasks other than transaction management, and their closed, non-modular system architecture [117]. Some of the above-mentioned ERP limitations are partly ameliorated by the usage of cloud-based ERP systems. However, cloud-based ERP systems present also limitations such as integration and functionality limitations as well as data migration, data integrity and security challenges [118, 119]. Arguably, legacy ERP systems present limited integration capabilities and may not have the ability to seamlessly integrate with other systems and technologies, a significant barrier hindering the flow of information across different parts of the SC. In addition, legacy ERP systems support inefficient data management schemes which are unable to effectively manage granular access control to SC data, leading to inefficiencies and difficulties in accessing and using relevant information, especially granular information within multi-entities. Besides, current SC networks face a multitude of challenges, especially with respect to proper information sharing in a multi-entity global SC environment while taking into account data integrity, fine-grained access control requirements and granularity in information sharing.

Blockchain technology has the potential to improve information sharing in the SC by providing a secure, decentralized platform for storing and sharing SC-related data. Based on the various challenges described above, we propose a blockchain-enabled high level architecture for enhancing information interoperability among multiple SC stakeholders by integrating ERP systems and blockchain technology. In particular, the proposed architecture enables the secure, fine-grained information sharing within the SC ecosystem, while taking into account information integrity guarantees, distributed trust management requirements, fine-grained cross domain access control policies and auditing features. The proposed blockchain architecture acts as a service layer on top of existing ERP systems to achieve fine-grained intra- and cross-domain access control. A private blockchain is combined with four fully functional Smart Contracts to enable access control and trust management services, a data handler service, ensuring data integrity for both insiders and outsiders, and an audit mechanism. The access control scheme employs a mechanism that grants access to users with certificates issued by different authorities, based on the various roles and access levels determined by the access policies agreed upon by all parties.

8.2 High-level Description of the Proposed Architecture

The proposed architecture aims to solve the problem of secure fine-grained information sharing in the Supply Chain ecosystem, which could significantly improve and automate several procedures and enhance decision-making, privacy and data integrity. It should be noted that the inherent complexities of today's business environment require system integration solutions that offer multi-domain granular access to data while taking into account distributed trust management prerequisites [120, 121, 122]. This proposed solution focuses specifically on the synergistic potential between blockchain and ERP integration. In particular, our system consists of a blockchain service layer that resides on top of multiple ERP systems using a fine-grained access mechanism and a trust management infrastructure responsible for handling certificates issued by different SC authorities. We have additionally incorporated an integrity control mechanism into the system to address the requirement for strong integrity guarantees, especially for data retrieved from external sources. This data in-

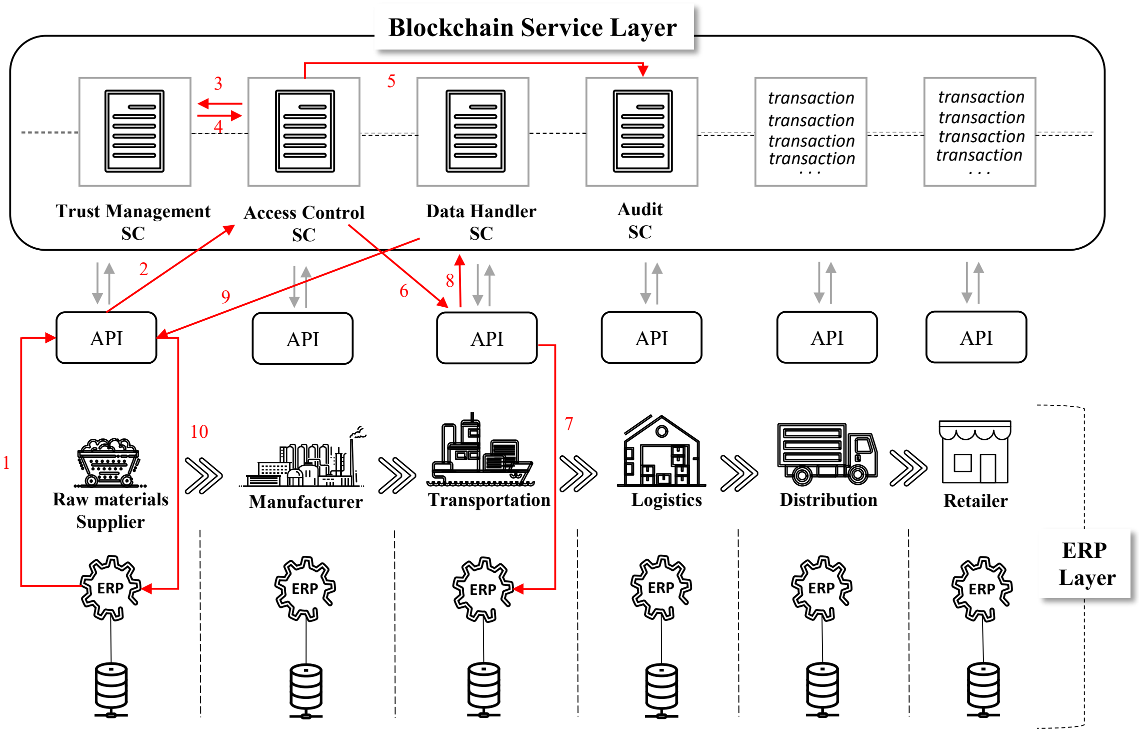


Figure 8.1: Blockchain service layer architecture

tegrity assurance mechanism was developed as a separate mechanism for increased usability, as it can function as a stand-alone service. The central component of this architecture (see Fig. 8.1) is a Hyperledger Fabric private blockchain that maintains four smart contracts for the various services. Note that we designed the architecture by considering the complexity of various SC types and the unique requirements for each case. In this sense, our model is generic and may require further refinement for each scenario.

8.3 Blockchain Layer Services and Actors

SC networks may include various types of organizations, each of which may have numerous types of users who must access various information types, both within the organization and the data shared by other organizations. In the current state of the SC ecosystem, users obtain a public/private key pair that has been validated by a trusted Public Key Infrastructure (PKI) and is used to provide granular access to each organization's ERP system. For instance, an employee responsible for a company's warehouse will be granted access to ERP data for the warehouse based on the attributes depicted on their certificate. A trust anchor for each type of certificate should exist on the blockchain to provide the same level of access for cross-domain data. In this way, if company A has issued a certificate for user X, then this certificate should also be verifiable from the system to control the access for shared data from company B. In the subsections below, we present the main services of the proposed architecture.

8.3.1 Interoperable and Decentralized Certificate Management

Each stakeholder should be able to independently manage the trust for their organization, while certificates issued by different authorities should be interoperable and accepted by all parties. Such an approach enables the development of certificate-based access policies for various user roles within and outside organisations. For instance, an accountant at organization A may require access to information stored in the database of organization B. To gain access, a valid certificate should be presented (which will be validated by the Trust Management mechanism) before the request is forwarded to the access control mechanism (which will grant him access according to the pre-defined access policies). In addition, trust management should be adaptive, allowing for the dynamic addition and removal of organizations. For instance, it must be possible to add a new stakeholder, assuming consent among existing stakeholders.

8.3.2 Inter-domain and Cross-domain Access Control

The system must ensure that all stakeholders in a specific Supply Chain network have granular access to inter-domain and cross-domain ERP data based on predefined access policies agreed upon at the role level by the stakeholders. For instance, a supplier of raw materials must be aware of the current stock of its materials in the warehouses of the factory it provides. However, some data must remain confidential and should not be shared outside the organization (for example, information regarding employees' personal data). Therefore, the system should guarantee that only specific types of data are shareable and only among those who have the permission rights (according to the policies) to do so.

8.3.3 Data Integrity Assurance

Database centralization offers several advantages concerning corporate data by providing a single point of contact and has been proven efficient over the years for data management inside an organization. At the same time, the fact that complete control over all processes is given to a database administrator is a critical drawback for information sharing across multiple organizations. False data retrieved from an external source may lead to erroneous decisions, highlighting the importance of integrity and reliability of mutually shared information. For instance, if a product is transported in unsuitable conditions, the manufacturer may recall it to preserve consumers' confidence. Additionally, the organization could file a claim for compensation with the transport company. In such cases, the participants may have conflicting interests, and one or more companies may wish to conceal the truth. From this perspective, the system must provide a service that enforces participants' trust in the accuracy and integrity of the provided data by preventing even the data owner from altering them.

8.3.4 Auditing

A blockchain stores all transactions in chronological order along the chain. To meet the needs of the Supply Chain, however, a mechanism is required to allow external users to verify the actions taken so that no one can later disclaim responsibility for them. The system should provide a service that permits the retrieval of transactional information at a granular level only for users with the required permissions.

8.4 Implementation

Following the discussion of the proposed architecture's requirements and services in sub-Section 8.3, in this subsection, we focus on the technical aspects of our blockchain service layer. Our blockchain is based on the Hyperledger Fabric platform v2.2 and utilizes smart contracts to automate the abovementioned processes. The system's architecture consists of a network of nodes running an Hyperledger Fabric client on each stakeholder. The nodes communicate with each other to propagate transactions and maintain the integrity of the blockchain. The consensus algorithm used is Raft[1], which allows for a decentralized and trustless system. Assuming that all users have obtained an attribute certificate from their organisation for accessing specific data according to their role, we used the Hyperledger Fabric CA v1.5.6 for managing all system certificates.

To facilitate the integration of our system with the underlying ERP systems of each stakeholder, we used an API that allows for easy communication with the blockchain (depicted in Fig. 8.1). Each stakeholder has its unique API, depending on the type of ERP used. The API components are used to add transactions, query the blockchain and interact with the smart contracts providing routing and forwarding for user requests. We assume that all data transmitted from the ERP systems are in $json$ format (the most common file format used by ERP systems). We have implemented the Hyperledger Fabric gateway v1.1 as the intermediate mechanism.

As shown in Fig. 8.2, each organization i transmits its root certificate, $R–Crt\_i$, to the blockchain during the initialization phase, where it is stored in the current block. Each organization is responsible for issuing employee certificates; only the root certificate is stored on the chain. In addition, while all shareable data are sent to an organization's ERP for storage, they are also hashed, $Hash(Data_i(k))$, and the output is sent to the blockchain, as a *commitment*, creating a link with the specific data based on a global id.

The access policies that define the access rights to the shared data are jointly agreed upon by the participants based on consensus and published on the blockchain. When the organizations need to change the access policy, a newer version is published on the blockchain, and a pointer is assigned to indicate the latest version.

All services at the blockchain layer are implemented through four smart contracts, self-

[1]https://raft.github.io/

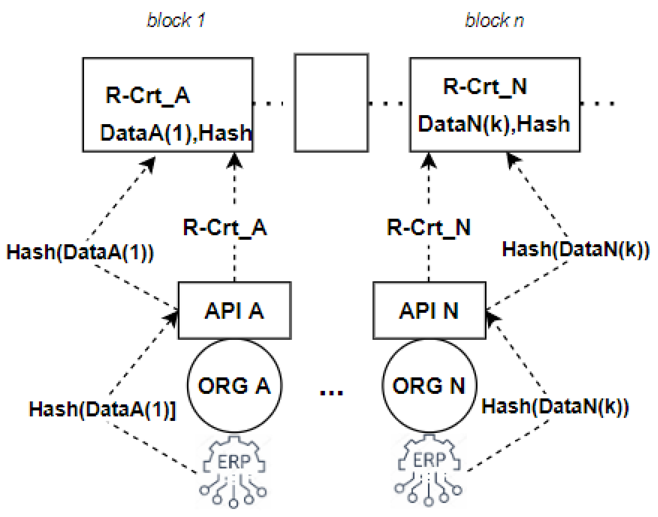


Figure 8.2: Storing root certificates and hashed data on the blockchain

executing programs published on the Blockchain, each of which incorporates a unique set of features. With smart contracts, we can only achieve the logical interconnection of independent ERP systems for this data that must be shared with other parties. In particular:

Trust Management Smart Contract (TMSC): integrates three services for updating root certificates, adding or removing a Certificate Authority and validating user certificates through the operation of the corresponding functions ($cert\_updt, add\_CA, remove\_CA, cert\_vld$).

Access Control Smart Contract (ACSC): is the first point of contact for every user who wants to access the shared data taking as input the $\{data\_id\}$ and the $\{usr\_crt\}$, which TMSC validates. Then, based on the assigned attributes, this smart contract enforces the predefined policy (following an access list published on the blockchain) and grants or denies access to the shared data. An updated policy can be published on the blockchain with the *upd_policy* function only if majority agreement is reached among the stakeholders by assigning an indicator for the most recent version. The request is sent to the appropriate API if the policy permits access.

Data Handler Smart Contract (DHSC): implements all services pertaining to the management of only-sharable data. When a new data registry is created on the ERP's database, the corresponding API calculates the hash, stores the data on the blockchain using the {*store_data*} and generates a logical array for mapping the data. When a user is granted access to data, the {*data_vld*} function is invoked to validate their integrity by recalculating their hash on-the-fly.

Audit Smart Contract (ASC): is responsible for maintaining an immutable global access log, provided that most stakeholders are trustworthy. The ASC is enabled whenever a new access request is processed. When a new user request is received, the ACSC calls the *store_log* function of the ASC to record a new transaction log on the blockchain containing the request's details (e.g. user-id, organization, data-id etc.). A *retrieve_log* function is also implemented, the activation of which requires the consensus of the majority of stakeholders to grant an external third party access to transaction logs.

Cross-domain Access to Shareable ERP Data: In Fig. 8.1 we have depicted in Red the basic steps made by the system when a user wants to retrieve data from an external ERP. As a first step, the user accesses the system by presenting an attribute certificate (issued independently by their organization). The certificate is received by the ACSC and is forwarded to TMSC for validation and attribute assignment (Steps 3 and 4). At the same time, ACSC triggers *store_log* function of the ASC for creating a new log. Then ACSC checks if the user has permission to access these data according to their attributes and the pre-defined policy and forwards the request (Org:A, Data_id: X) to the appropriate API Step 6). Before the data is sent back to the user, they are hashed on-the-fly, and the output is compared with the hash stored on the chain (Step 8) to ensure integrity. Finally, the requested data reach the user through the API of their organization (Step 9 and 10).

SECTION IV
Conclusions and Future work

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

In this Chapter, we synthesize the key insights and contributions derived from our extensive research on distributed security and trust management in MA and MD environments with a specific focus in healthcare and supply chain management systems. This thesis aimed to address significant knowledge gaps in the fields of trust management, access control and secure data management within complex, multifaceted ecosystems by providing innovative solutions that advance the current state of the art and offer practical applications for real-world challenges.

9.1 Concluding Remarks

Our research commenced with a thorough examination of the foundational principles in blockchain technology, particularly focusing on the functionalities and implications of Smart Contracts within Ethereum and Hyperledger networks and also the related work for distributed trust management, fine-grained access control and encryption mechanisms (Chapters 1 and 2). By establishing this groundwork, we set the stage for the introduction of our novel Hierarchical Multi-Blockchain Access Control Model (HMBAC) in Chapter 3. This model addresses the complexities of managing access in environments governed by multiple authorities and domains, leveraging the decentralized nature of blockchain to enhance efficiency, scalability, and security.

The architectural design of the hierarchical multi-blockchain framework, detailed in Chapter 4, builds upon the HMBAC model. This design incorporates fundamental security features and outlines the system's principal components, which are crucial for ensuring robust access control operations. Chapter 5 further elaborates on the implementation details, providing an in-depth analysis of the relevant APIs and Smart Contracts that underpin the framework's functionality.

Chapter 6 presents a comprehensive evaluation of the proposed framework's security and performance. Through rigorous scalability and benchmarking analyses, we demonstrate the system's operational efficiency and its capacity to manage access requests across various configurations. These results validate the effectiveness of our approach in addressing the challenges inherent in complex multi-domain environments.

Beyond the primary focus on hierarchical access control, our research also explores additional security features. In Chapter 7, we propose a self-sovereign identity management subsystem designed to autonomously handle device IDs in distributed environments, thereby enhancing the framework's identity management capabilities. Chapter 8 introduces a novel architecture that integrates blockchain technology with legacy Enterprise Resource Planning (ERP) systems, facilitating improved data interconnection and trust among stakeholders.

Our research identified and addressed several critical gaps in the existing body of knowledge regarding trust and security in multi-authority and multi-domain environments. In particular:

- *Secure Interoperability of Trust Infrastructures:* Traditional trust models struggle with interoperability in environments with multiple authorities and domains. We addressed this gap by proposing a system that employs multiple blockchains and cryptographic mechanisms to ensure seamless and secure interoperability. This system allows for the distributed and self-sustaining management of trust, enabling credential interoperability without a globally trusted root authority. Details of this solution are discussed in Chapters 3 and 4.

- *Fine-Grained Access, Privacy-Preserving Encryption, and Immutable Logging:* Balancing fine-grained access control with privacy and immutable logging is challenging. We proposed an integrated solution combining attribute-based encryption (ABE) and blockchain technology to ensure granular access control while preserving privacy and maintaining a tamper-proof audit trail. This approach is thoroughly analyzed in Chapters 5 and 6.

- *Self-Managed Identities and Credentials:* Centralized identity management systems pose significant risks in multi-authority environments. Our contribution includes a model for self-managed identities based on decentralized identity technologies, such as blockchain and decentralized identifiers (DIDs). This model supports secure and autonomous identity management, enhancing privacy and control. Chapter 7 provides an in-depth exploration of this model.

- *Interconnection with Legacy Systems:* The integration of new decentralized systems with existing ERP systems poses significant challenges. We developed a blockchain-based architecture that integrates seamlessly with legacy ERP systems, mitigating concerns about costs and infrastructure overhaul. This strategic integration is detailed in Chapter 8.

In conclusion, this thesis has made significant strides in advancing the field of blockchain-based access control by introducing innovative models, architectures, and implementations. Our research provides a solid foundation for future studies and practical applications, aiming to harness the full potential of blockchain technology in managing access and security within multifaceted ecosystems, that have special requirements both in terms of operation and security. The proposed hierarchical multi-blockchain framework and its complementary subsystems represent a significant contribution to both academic research and industry practices, paving the way for more secure and efficient digital infrastructures.

9.2 Limitations and Future Work

The findings of this research have significant implications for the development and deployment of secure, scalable, and efficient blockchain-based access control systems. By addressing the identified knowledge gaps and presenting practical solutions, our work contributes to the advancement of blockchain technology in managing complex ecosystems.

However, our study also encountered certain limitations that warrant further investigation. For instance, the scalability of our framework in extremely large-scale environments and the integration of additional privacy-preserving mechanisms are areas that require deeper exploration. Moreover, the dynamic nature of blockchain technology and its rapidly evolving landscape present ongoing challenges and opportunities for future research.

In light of these considerations, we propose several directions for future work:

- **Extending the capabilities of our hierarchical multi-blockchain framework:** In the context of this PhD thesis, the medical and supply chain ecosystems were investigated. Supporting more diverse and complex use cases would enhance the framework's applicability. By accommodating a wider range of applications, the framework can become more versatile and robust.

- **Exploring advanced cryptographic techniques and integrating machine learning algorithms for predictive access control:** In the context of this PhD thesis, we modified the MA-CP-ABE scheme of [97], to achieve fine-grained access control. Incorporating cutting-edge cryptographic methods and leveraging machine learning can further strengthen the system's security and performance. These techniques can provide more sophisticated and adaptive security measures.

- **Investigating the socio-economic impacts of deploying such frameworks in real-world scenarios:** Understanding the broader implications of blockchain deployment, including social and economic effects, will provide valuable insights that can guide future implementations and policy-making.

- **Hardware and Software Trusted Computing (Hardware Immutability):** The proposed framework presented in this thesis should be combined with trusted hardware components in order to achieve an ever greater level of security. Future research should focus on developing and integrating trusted computing bases that guarantee hardware integrity and prevent tampering. This includes exploring hardware-based security modules and secure boot processes to create a more secure foundation for blockchain operations.

- **Extension of SSI in the Metaverse:** Future research on the implementation of Self-Sovereign Identity (SSI) in the metaverse holds significant promise for advancing digital identity management in immersive virtual environments. Research could explore innovative approaches to integrating SSI with augmented reality (AR) and virtual reality (VR) technologies, focusing on enhancing privacy and trust in digital interactions. Additionally, investigating the scalability of SSI solutions to support millions of users and devices in a decentralized manner will be crucial. This research could also delve into developing advanced cryptographic techniques, such as homomorphic encryption and decentralized identifiers (DIDs), to further enhance the security and functionality of SSI in the metaverse, ultimately paving the way for a more secure and user-centric digital future.

- **Interoperability with Different Blockchains:** Achieving seamless interoperability between different blockchain networks remains a significant challenge. Future work should investigate standardized protocols and frameworks that enable diverse blockchains to communicate and transact with each other effectively. This includes exploring cross-chain communication protocols, atomic swaps, and blockchain bridges to facilitate interoperability and enhance the overall ecosystem's functionality.

By tackling these challenges, future work can build upon the foundations laid in this study, ultimately leading to more robust, secure, and versatile solutions. The continuous evolution of this research will not only enhance our understanding but also contribute to the practical applications and real-world impact of enhanced security in multi authority and multi domain environments.

REFERENCES

[1] V. Malamas, T. Dasaklis, P. Kotzanikolaou, M. Burmester, and S. Katsikas, "A forensics-by-design management framework for medical devices based on blockchain," in *2019 IEEE world congress on services (SERVICES)*, vol. 2642. IEEE, 2019, pp. 35–40.

[2] V. Malamas, F. Chantzis, T. K. Dasaklis, G. Stergiopoulos, P. Kotzanikolaou, and C. Douligeris, "Risk assessment methodologies for the internet of medical things: A survey and comparative appraisal," *IEEE Access*, vol. 9, pp. 40 049–40 075, 2021.

[3] D. Koutras, V. Malamas, P. Kotzanikolaou, and T. Dasaklis, "A risk assessment methodology for supply chain tracking services," in *2023 International Conference On Cyber Management And Engineering (CyMaEn)*. IEEE, 2023, pp. 555–559.

[4] V. Malamas, G. Palaiologos, P. Kotzanikolaou, M. Burmester, and D. Glynos, "Janus: Hierarchical multi-blockchain-based access control (hmbac) for multi-authority and multi-domain environments," *Applied Sciences*, vol. 13, no. 1, p. 566, 2022.

[5] V. Malamas, P. Kotzanikolaou, T. K. Dasaklis, and M. Burmester, "A hierarchical multi blockchain for fine grained access to medical data," *IEEE Access*, vol. 8, pp. 134 393–134 412, 2020.

[6] F. Fotopoulos, V. Malamas, T. K. Dasaklis, P. Kotzanikolaou, and C. Douligeris, "A blockchain-enabled architecture for iomt device authentication," in *2020 IEEE Eurasia conference on IoT, communication and engineering (ECICE)*. IEEE, 2020, pp. 89–92.

[7] V. Malamas, D. Koutras, and P. Kotzanikolaou, "Uninterrupted trust: Continuous authentication in blockchain-enhanced supply chains," in *8th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM 2023)*. IEEE, 2023.

[8] V. Malamas, T. K. Dasaklis, T. G. Voutsinas, and P. Kotzanikolaou, "Blockchain service layer for erp data interoperability among multiple supply chain stakeholders," in *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, 2023, pp. 145–150.

[9] T. K. Dasaklis and V. Malamas, "A review of the lightning network's evolution: Unraveling its present state and the emergence of disruptive digital business models," *Journal of Theoretical and Applied Electronic Commerce Research*, vol. 18, no. 3, pp. 1338–1364, 2023.

[10] V. Malamas, T. K. Dasaklis, V. Arakelian, and G. Chondrokoukis, "A blockchain framework for digitizing securities issuance: the case of green bonds," *Journal of Sustainable Finance & Investment*, pp. 1–27, 2023.

[11] S. Haber and W. S. Stornetta, *How to time-stamp a digital document.* Springer, 1991.

[12] D. Bayer, S. Haber, and W. S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," in *Sequences II: Methods in Communication, Security, and Computer Science.* Springer, 1993, pp. 329–334.

[13] N. S. Bitcoin, "Bitcoin: A peer-to-peer electronic cash system," 2008.

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

[15] N. Lachtar, A. A. Elkhail, A. Bacha, and H. Malik, "A cross-stack approach towards defending against cryptojacking," *IEEE Computer Architecture Letters*, vol. 19, no. 2, pp. 126–129, 2020.

[16] S. S. Sabry, N. M. Kaittan, and I. Majeed, "The road to the blockchain technology: Concept and types," *Periodicals of Engineering and Natural Sciences*, vol. 7, no. 4, pp. 1821–1832, 2019.

[17] A. R. Sai, J. Buckley, B. Fitzgerald, and A. Le Gear, "Taxonomy of centralization in public blockchain systems: A systematic literature review," *Information Processing & Management*, vol. 58, no. 4, p. 102584, 2021.

[18] R. Yang, R. Wakefield, S. Lyu, S. Jayasuriya, F. Han, X. Yi, X. Yang, G. Amarasinghe, and S. Chen, "Public and private blockchain in construction business process and information integration," *Automation in construction*, vol. 118, p. 103276, 2020.

[19] A. Alkhateeb, C. Catal, G. Kar, and A. Mishra, "Hybrid blockchain platforms for the internet of things (iot): A systematic literature review," *Sensors*, vol. 22, no. 4, p. 1304, 2022.

[20] Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," *Future Generation Computer Systems*, vol. 105, pp. 475–491, 2020.

[21] S. Wang, Y. Yuan, X. Wang, J. Li, R. Qin, and F.-Y. Wang, "An overview of smart contract: architecture, applications, and future trends," in *2018 IEEE Intelligent Vehicles Symposium (IV).* IEEE, 2018, pp. 108–113.

[22] A. Tobin and D. Reed, "The inevitable rise of self-sovereign identity," *The Sovrin Foundation*, vol. 29, no. 2016, p. 18, 2016.

[23] K. Cameron, R. Posch, and K. Rannenberg, "Appendix d. proposal for a common identity framework: A user-centric identity metasystem," *The Future of Identity in the Information Society*, p. 477, 2009.

[24] Y. Chen, M. Lei, W. Ren, Y. Ren, and Z. Qu, "RoFa: A Robust and Flexible Fine-Grained Access Control Scheme for Mobile Cloud and IoT based Medical Monitoring," *Fundamenta Informaticae*, vol. 157, no. 1-2, pp. 167–184, 2018.

[25] M. F. F. Khan and K. Sakamura, “Fine-grained access control to medical records in digital healthcare enterprises,” in *2015 International Symposium on Networks, Computers and Communications (ISNCC)*, 2015, pp. 1–6.

[26] E. Mrema and V. Kumar, “Fine Grained Attribute Based Access Control of Healthcare Data,” in *12th International Symposium on Medical Information and Communication Technology (ISMICT)*, 2018, pp. 1–5.

[27] Y. Zhao, P. Fan, H. Cai, Z. Qin, and H. Xiong, “Attribute-based encryption with non-monotonic access structures supporting fine-grained attribute revocation in m-healthcare,” *International Journal of Network Security*, vol. 19, no. 6, pp. 1044–1052, 2017.

[28] S. Roy, A. K. Das, S. Chatterjee, N. Kumar, S. Chattopadhyay, and J. J. P. C. Rodrigues, “Provably Secure Fine-Grained Data Access Control over Multiple Cloud Servers in Mobile Cloud Computing Based Healthcare Applications,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 457–468, 2019.

[29] H. Balinsky and N. Mohammad, “Fine grained access of interactive personal health records,” in *2015 ACM Symposium on Document Engineering*, 2015, pp. 207–210.

[30] M. K. Debnath, S. Samet, and K. Vidyasankar, “A secure revocable personal health record system with policy-based fine-grained access control,” in *13th Annual Conference on Privacy, Security and Trust (PST)*, 2015, pp. 109–116.

[31] C. Gritti, W. Susilo, T. Plantard, K. Liang, and D. S. Wong, “Empowering personal health records with cloud computing: How to encrypt with forthcoming fine-grained policies efficiently,” *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 5, no. 4, pp. 3–28, 2014.

[32] M. Li, S. Yu, K. Ren, and W. Lou, “Securing personal health records in cloud computing: Patient-centric and fine-grained data access control in multi-owner settings,” in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, 2010, vol. 50 LNICST, pp. 89–106.

[33] W. Li, B. M. Liu, D. Liu, R. P. Liu, P. Wang, S. Luo, and W. Ni, “Unified Fine-grained Access Control for Personal Health Records in Cloud Computing,” *IEEE Journal of Biomedical and Health Informatics*, 2018.

[34] Y. Liu, Y. Zhang, J. Ling, and Z. Liu, “Secure and fine-grained access control on e-healthcare records in mobile cloud computing,” *Future Generation Computer Systems*, vol. 78, pp. 1020–1026, 2018.

[35] L. Mohanan and A. B. Varghese, “Flexible, scalable and fine grained access control for medical data in cloud using attribute based encryption,” *International Journal of Applied Engineering Research*, vol. 10, no. 23, pp. 43 378–43 383, 2015.

[36] W. Zhang, Y. Lin, J. Wu, and T. Zhou, "Inference Attack-Resistant E-Healthcare Cloud System with Fine-Grained Access Control," *IEEE Transactions on Services Computing*, 2018.

[37] P. T. B. Hue, S. Wohlgemuth, I. Echizen, D. T. B. Thuy, and N. D. Thuc, "Fine-grained access control for electronic health record systems," in *International Conference on U- and E-Service, Science and Technology*, 2010, pp. 31–38.

[38] S. Pal, M. Hitchens, V. Varadharajan, and T. Rabehaja, "On design of a fine-grained access control architecture for securing iot-enabled smart healthcare systems," in *14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2017, pp. 432–441.

[39] X. Wang, L. Wang, Y. Li, and K. Gai, "Privacy-Aware Efficient Fine-Grained Data Access Control in Internet of Medical Things Based Fog Computing," *IEEE Access*, vol. 6, pp. 47 657–47 665, 2018.

[40] L. Y. Yeh, P. Y. Chiang, Y. L. Tsai, and J. L. Huang, "Cloud-Based Fine-Grained Health Information Access Control Framework for LightweightIoT Devices with Dynamic Auditing andAttribute Revocation," *IEEE Transactions on Cloud Computing*, vol. 6, no. 2, pp. 532–544, 2018.

[41] H. Ma, R. Zhang, G. Yang, Z. Song, K. He, and Y. Xiao, "Efficient Fine-Grained Data Sharing Mechanism for Electronic Medical Record Systems with Mobile Devices," *IEEE Transactions on Dependable and Secure Computing*, 2018.

[42] L. Selvam and R. J. Arokia, "Secure Data Sharing of Personal Health Records in Cloud Using Fine-Grained and Enhanced Attribute-Based Encryption," in *Proceedings of the 2018 International Conference on Current Trends towards Converging Technologies, ICCTCT 2018*, 2018.

[43] C. Guo, R. Zhuang, Y. Jie, Y. Ren, T. Wu, and K. K. R. Choo, "Fine-grained Database Field Search Using Attribute-Based Encryption for E-Healthcare Clouds," *Journal of Medical Systems*, vol. 40, no. 11, 2016.

[44] Z. Chen, F. Zhang, P. Zhang, J. K. Liu, J. Huang, H. Zhao, and J. Shen, "Verifiable keyword search for secure big data-based mobile healthcare networks with fine-grained authorization control," *Future Generation Computer Systems*, vol. 87, pp. 712–724, 2018.

[45] J. Sun, X. Wang, S. Wang, and L. Ren, "A searchable personal health records framework with fine-grained access control in cloud-fog computing," *PLoS ONE*, vol. 13, no. 11, 2018.

[46] T. McGhin, K. K. R. Choo, C. Z. Liu, and D. He, "Blockchain in healthcare applications: Research challenges and opportunities," *Journal of Network and Computer Applications*, vol. 135, pp. 62–75, 2019.

[47] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, “MedRec: Using blockchain for medical data access and permission management,” in *Proceedings - 2016 2nd International Conference on Open and Big Data, OBD 2016*. IEEE, 8 2016, pp. 25–30.

[48] A. Al Omar, M. S. Rahman, A. Basu, and S. Kiyomoto, “MediBchain: A Blockchain Based Privacy Preserving Platform for Healthcare Data,” in *Security, Privacy and Anonymity in Computation, Communication and Storage*, 2017, pp. 534–543.

[49] T. Mikula and R. H. Jacobsen, “Identity and Access Management with Blockchain in Electronic Healthcare Records,” in *2018 21st Euromicro Conference on Digital System Design (DSD)*. IEEE, 8 2018, pp. 699–706.

[50] Y. Du, J. Liu, Z. Guan, and H. Feng, “A medical information service platform based on distributed cloud and blockchain,” in *Proceedings - 3rd IEEE International Conference on Smart Cloud, SmartCloud 2018*, 2018, pp. 34–39.

[51] A. F. Hussein, N. ArunKumar, G. Ramirez-Gonzalez, E. Abdulhay, J. M. R. S. Tavares, and V. H. C. de Albuquerque, “A medical records managing and securing blockchain based system supported by a Genetic Algorithm and Discrete Wavelet Transform,” *Cognitive Systems Research*, vol. 52, pp. 1–11, 2018.

[52] Y. Chen, S. Ding, Z. Xu, H. Zheng, and S. Yang, “Blockchain-Based Medical Records Secure Storage and Medical Service Framework,” *Journal of Medical Systems*, vol. 43, no. 1, 2018.

[53] X. Zhang and S. Poslad, “Blockchain Support for Flexible Queries with Granular Access Control to Electronic Medical Records (EMR),” in *IEEE International Conference on Communications*, vol. 2018-May, 2018.

[54] K. Al Nuaimi, N. Mohamed, M. Al Nuaimi, and J. Al-Jaroodi, “A survey of load balancing in cloud computing: Challenges and algorithms,” in *2012 second symposium on network cloud computing and applications*. IEEE, 2012, pp. 137–142.

[55] M. P. Andersen, S. Kumar, M. AbdelBaky, G. Fierro, J. Kolb, H.-S. Kim, D. E. Culler, and R. A. Popa, “{WAVE}: A decentralized authorization framework with transitive delegation,” in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 1375–1392.

[56] H. Shafagh, L. Burkhalter, S. Ratnasamy, and A. Hithnawi, “Droplet: Decentralized authorization and access control for encrypted data streams,” in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 2469–2486.

[57] A. R. Rajput, Q. Li, M. T. Ahvanooey, and I. Masood, “Eacms: Emergency access control management system for personal health record based on blockchain,” *IEEE Access*, vol. 7, pp. 84 304–84 317, 2019.

[58] A. S. Shahraki, C. Rudolph, and M. Grobler, "A dynamic access control policy model for sharing of healthcare data in multiple domains," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE).* IEEE, 2019, pp. 618–625.

[59] Y. Miao, X. Liu, K.-K. R. Choo, R. H. Deng, J. Li, H. Li, and J. Ma, "Privacy-preserving attribute-based keyword search in shared multi-owner setting," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[60] Y. Xu, X. Dong, and J. Shen, "Multi-authority attribute-based encryption supporting hierarchal access policy and range policy," in *2020 International Conference on Computer Communication and Network Security (CCNS).* IEEE, 2020, pp. 81–86.

[61] M. Xiao and X. Hu, "Multi-authority attribute-based encryption access control scheme in wireless body area network," in *2018 3rd International Conference on Information Systems Engineering (ICISE).* IEEE, 2018, pp. 39–45.

[62] Z. Zhang and S. Zhou, "A decentralized strongly secure attribute-based encryption and authentication scheme for distributed internet of mobile things," *Computer Networks*, vol. 201, p. 108553, 2021.

[63] R. Sarma, C. Kumar, and F. A. Barbhuiya, "Macfi: A multi-authority access control scheme with efficient ciphertext and secret key size for fog-enhanced iot," *Journal of Systems Architecture*, vol. 123, p. 102347, 2022.

[64] H. Guo, E. Meamari, and C.-C. Shen, "Multi-authority attribute-based access control with smart contract," in *Proceedings of the 2019 international conference on blockchain technology*, 2019, pp. 6–11.

[65] S. Das and S. Namasudra, "Multi-authority cp-abe-based access control model for iot-enabled healthcare infrastructure," *IEEE Transactions on Industrial Informatics*, 2022.

[66] C. Liu, F. Xiang, and Z. Sun, "Multiauthority attribute-based access control for supply chain information sharing in blockchain," *Security and Communication Networks*, vol. 2022, 2022.

[67] Q. Li, H. Zhu, J. Xiong, R. Mo, Z. Ying, and H. Wang, "Fine-grained multi-authority access control in iot-enabled mhealth," *Annals of Telecommunications*, vol. 74, no. 7, pp. 389–400, 2019.

[68] Q. Xu, C. Tan, Z. Fan, W. Zhu, Y. Xiao, and F. Cheng, "Secure multi-authority data access control scheme in cloud storage system based on attribute-based signcryption," *IEEE Access*, vol. 6, pp. 34 051–34 074, 2018.

[69] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *Proceedings of the 17th ACM conference on Computer and communications security*, 2010, pp. 735–737.

[70] Z. Wan, R. H. Deng *et al.*, "Hasbe: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE transactions on information forensics and security*, vol. 7, no. 2, pp. 743–754, 2011.

[71] M. Ali, J. Mohajeri, M.-R. Sadeghi, and X. Liu, "A fully distributed hierarchical attribute-based encryption scheme," *Theoretical Computer Science*, vol. 815, pp. 25–46, 2020.

[72] K. Riad, T. Huang, and L. Ke, "A dynamic and hierarchical access control for iot in multi-authority cloud storage," *Journal of Network and Computer Applications*, vol. 160, p. 102633, 2020.

[73] L. Bai, K. Fan, Y. Bai, X. Cheng, H. Li, and Y. Yang, "Cross-domain access control based on trusted third-party and attribute mapping center," *Journal of Systems Architecture*, vol. 116, p. 101957, 2021.

[74] K. Gai, J. Guo, L. Zhu, and S. Yu, "Blockchain meets cloud computing: A survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2009–2030, 2020.

[75] I. Riabi, H. K. B. Ayed, and L. A. Saidane, "A survey on blockchain based access control for internet of things," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, 2019, pp. 502–507.

[76] H. Li, L. Pei, D. Liao, S. Chen, M. Zhang, and D. Xu, "Fadb: A fine-grained access control scheme for vanet data based on blockchain," *IEEE Access*, vol. 8, pp. 85 190–85 203, 2020.

[77] I. Sukhodolskiy and S. Zapechnikov, "A blockchain-based access control system for cloud storage," in *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*. IEEE, 2018, pp. 1575–1578.

[78] S. Wang, Y. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," *Ieee Access*, vol. 6, pp. 38 437–38 450, 2018.

[79] C. Yang, L. Tan, N. Shi, B. Xu, Y. Cao, and K. Yu, "Authprivacychain: A blockchain-based access control framework with privacy protection in cloud," *IEEE Access*, vol. 8, pp. 70 604–70 615, 2020.

[80] S. Banerjee, B. Bera, A. K. Das, S. Chattopadhyay, M. K. Khan, and J. J. Rodrigues, "Private blockchain-envisioned multi-authority cp-abe-based user access control scheme in iiot," *Computer Communications*, vol. 169, pp. 99–113, 2021.

[81] F. Marino, C. Moiso, and M. Petracca, "Pkiot: A public key infrastructure for the internet of things," *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 10, p. e3681, 2019.

[82] A. Herzberg, Y. Mass, J. Mihaeli, D. Naor, and Y. Ravid, "Access control meets public key infrastructure, or: Assigning roles to strangers," in *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*. IEEE, 2000, pp. 2–14.

[83] G. Antonio, M. Gregorio, and C. Oscar, "New security services based on pki," *Future Generation Computer Systems*, vol. 19, no. 2, pp. 251–262, 2003.

[84] J. Sun and Y. Fang, "Cross-domain data sharing in distributed electronic health record systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 6, pp. 754–764, 2009.

[85] Z. Qikun, G. Yong, Z. Quanxin, W. Ruifang, and T. Yu-An, "A dynamic and cross-domain authentication asymmetric group key agreement in telemedicine application," *IEEE Access*, vol. 6, pp. 24 064–24 074, 2018.

[86] T. D. Nguyen, M. A. Gondree, D. J. Shifflett, J. Khosalim, T. E. Levin, and C. E. Irvine, "A cloud-oriented cross-domain security architecture," in *2010-MILCOM 2010 MILITARY COMMUNICATIONS CONFERENCE.* IEEE, 2010, pp. 441–447.

[87] F. Tong, X. Chen, K. Wang, and Y. Zhang, "Ccap: A complete cross-domain authentication based on blockchain for internet of things," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3789–3800, 2022.

[88] H. Zhang and F. Zhao, "Cross-domain identity authentication scheme based on blockchain and pki system," *High-Confidence Computing*, vol. 3, no. 1, p. 100096, 2023.

[89] M. Shen, H. Liu, L. Zhu, K. Xu, H. Yu, X. Du, and M. Guizani, "Blockchain-assisted secure device authentication for cross-domain industrial iot," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 942–954, 2020.

[90] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, pp. 169–178.

[91] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, 2011, pp. 113–124.

[92] Y. Zhao, Y. Fan, and X. Bian, "Oo-ma-kp-abe-crf: Online/offline multi-authority key-policy attribute-based encryption with cryptographic reverse firewall for physical ability data," *Mathematics*, vol. 11, no. 15, p. 3333, 2023.

[93] C. Zhang, Z. Wang, L. Liu, G. Li, and H. Li, "A decentralized multi-authority attribute-based encryption scheme via blockchain for smart grid," in *2023 IEEE International Conference on Electrical, Automation and Computer Engineering (ICEACE).* IEEE, 2023, pp. 269–274.

[94] Y. Yao, H. Chen, L. Shen, K. Wang, and Q. Wang, "A cp-abe scheme based on lattice lwe and its security analysis," *Applied Sciences*, vol. 13, no. 14, p. 8043, 2023.

[95] S. Abdollahi, J. Mohajeri, and M. Salmasizadeh, "Highly efficient and revocable cp-abe with outsourcing decryption for iot," in *2021 18th International ISC Conference on Information Security and Cryptology (ISCISC).* IEEE, 2021, pp. 81–88.

[96] M. Chase, “Multi-authority attribute based encryption,” in *Theory of cryptography conference.* Springer, 2007, pp. 515–534.

[97] A. Lewko and B. Waters, “Decentralizing attribute-based encryption,” in *Annual international conference on the theory and applications of cryptographic techniques.* Springer, 2011, pp. 568–588.

[98] H. Qian, J. Li, Y. Zhang, and J. Han, “Privacy-preserving personal health record using multi-authority attribute-based encryption with revocation,” *International Journal of Information Security*, vol. 14, no. 6, pp. 487–497, 2015.

[99] Y. Rouselakis and B. Waters, “Efficient statically-secure large-universe multi-authority attribute-based encryption,” in *International Conference on Financial Cryptography and Data Security.* Springer, 2015, pp. 315–332.

[100] D. Ramesh and R. Priya, “Multi-authority scheme based cp-abe with attribute revocation for cloud data storage,” in *2016 International Conference on Microelectronics, Computing and Communications (MicroCom).* IEEE, 2016, pp. 1–4.

[101] Z. Zhang, C. Li, B. B. Gupta, and D. Niu, “Efficient compressed ciphertext length scheme using multi-authority cp-abe for hierarchical attributes,” *IEEE Access*, vol. 6, pp. 38 273–38 284, 2018.

[102] Electron, ““electronjs”,” https://www.electronjs.org/, accessed: 2022-12-16.

[103] D. Ongaro and J. Ousterhout, “In search of an understandable consensus algorithm,” in *2014 USENIX Annual Technical Conference (Usenix ATC 14)*, 2014, pp. 305–319.

[104] V. Malamas, G. Palaiologos, P. Kotzanikolaou, M. Burmester, and D. Glynos, “Janus,” https://census-labs.com/news/2022/06/21/janus-hmbac/, 2022.

[105] ““kubernetes”,” https://kubernetes.io/, accessed: 2022-12-16.

[106] Hashicorp, ““hashicorp vault”,” https://www.vaultproject.io/, accessed: 2022-12-16.

[107] ““rabbitmq”,” https://www.rabbitmq.com/, accessed: 2022-12-16.

[108] S. Edward, D. Czarnota, R. Tonic, and B. Perez, “Kubernetes security whitepaper,” *Kubernetes Security WG*, 2019.

[109] S. Mauw and M. Oostdijk, “Foundations of attack trees,” in *International Conference on Information Security and Cryptology.* Springer, 2005, pp. 186–198.

[110] B. Schneier, “Attack trees,” *Dr. Dobb’s journal*, vol. 24, no. 12, pp. 21–29, 1999.

[111] H. Forslund, “ERP systems’ capabilities for supply chain performance management,” *Industrial Management & Data Systems*, vol. 110, no. 3, pp. 351–367, 2010. [Online]. Available: https://doi.org/10.1108/02635571011030024

[112] M. F. Acar, S. Zaim, M. Isik, and F. Calisir, "Relationships among ERP, supply chain orientation and operational performance," *Benchmarking: An International Journal*, vol. 24, no. 5, pp. 1291–1308, 2017. [Online]. Available: https://doi.org/10.1108/BIJ-11-2015-0116

[113] P. Kelle and A. Akbulut, "The role of erp tools in supply chain information sharing, cooperation, and cost optimization," *International Journal of Production Economics*, vol. 93-94, pp. 41–52, 2005, proceedings of the Twelfth International Symposium on Inventories.

[114] S. E. Chang and Y. Chen, "When blockchain meets supply chain: A systematic literature review on current development and potential applications," *IEEE Access*, vol. 8, pp. 62 478–62 494, 2020.

[115] M. Pournader, Y. Shi, S. Seuring, and S. L. Koh, "Blockchain applications in supply chains, transport and logistics: a systematic review of the literature," *International Journal of Production Research*, vol. 58, no. 7, pp. 2063–2081, 2020.

[116] T. K. Dasaklis, T. G. Voutsinas, and A. Mihiotis, "Integrating blockchain with enterprise resource planning systems: Benefits and challenges," in *25th Pan-Hellenic Conference on Informatics*, ser. PCI 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 265–270.

[117] H. A. Akkermans, P. Bogerd, E. Yücesan, and L. N. van Wassenhove, "The impact of erp on supply chain management: Exploratory findings from a european delphi study," *European Journal of Operational Research*, vol. 146, no. 2, pp. 284–301, 2003.

[118] M. A. Abd Elmonem, E. S. Nasr, and M. H. Geith, "Benefits and challenges of cloud erp systems – a systematic literature review," *Future Computing and Informatics Journal*, vol. 1, no. 1, pp. 1–9, 2016.

[119] V. U. Sørheller, E. J. Høvik, E. Hustad, and P. Vassilakopoulou, "Implementing cloud erp solutions: a review of sociotechnical concerns," *Procedia Computer Science*, vol. 138, pp. 470–477, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918316983

[120] V. Malamas, G. Palaiologos, P. Kotzanikolaou, M. Burmester, and D. Glynos, "Janus: Hierarchical multi-blockchain-based access control (hmbac) for multi-authority and multi-domain environments," *Applied Sciences*, vol. 13, no. 1, p. 566, 2023.

[121] T. K. Agrawal, V. Kumar, R. Pal, L. Wang, and Y. Chen, "Blockchain-based framework for supply chain traceability: A case example of textile and clothing industry," *Computers & industrial engineering*, vol. 154, p. 107130, 2021.

[122] M. Wang, L. Rui, Y. Yang, Z. Gao, and X. Chen, "A blockchain-based multi-ca cross-domain authentication scheme in decentralized autonomous network," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2664–2676, 2022.