Department of Information Engineering and Computer Science

Bachelor's Degree in
Computer, Communication and Electronic Engineering

Final Dissertation

End-to-End Protection for Data Sharing Among Organizations in Cloud-Managed Blockchain Applications

Supervisor

PROF. SILVIO RANISE

Student

LUCA CLAUS

Co-Supervisor

DR. STEFANO BERLATO

DR. RICCARDO LONGO

Accademic year 2023/2024

Acknowledgments

I would like to begin by sincerely thanking Prof. Silvio Ranise for giving me the opportunity to write this thesis. I am immensely grateful to have been able to undertake the internship and thesis-writing period in such a prestigious environment as Fondazione Bruno Kessler (FBK), which allowed me to explore new fields and develop an invaluable set of technical skills. Special thanks go to Dr. Stefano Berlato e to Dr. Riccardo Longo, who have guided me throughout this wonderful journey, offering support from beginning to end. Thanks to their experience, wisdom, and dedication, I was able to explore new topics in a field where my knowledge was almost basic, to the point of learning and applying them in my thesis. No scientist has yet managed to calculate the amount of patience you have had with me, and for this, I am infinitely grateful and wish you all the best for the future. I would also like to thank the University of Trento, DISI department and all the professors who have enabled me to achieve this goal and to earn this degree.

I would also like to thank my brother Carlo for making my university journey possible by providing me with the necessary help and teaching me the (mathematical) foundations that were, unfortunately, lacking, to help me tackle the initial exams. Without you, I think it would have been quite difficult to complete this journey. Thank you.

Then, I want to thank my lifelong friends: Sepp, Pie, Cristl, Mena, Pippo e lastly Ari e Uoua :), who have always been there for anything (even if it was just to tease me); the best roommates in the world: Slanderlor (even though he does nothing) and chef Pippo; then Zano, with whom I spent 14 years together at school, and without whom these 3 years at university would have been empty, and also my study companions at BUC: Bertol and Zanet. There would be an endless list of friends, relatives and people I respect whom I would like to mention, but unfortunately, ink is expensive, and those who know me understand. Anyway, I want to thank all of you who are on this list, and I sincerely hope we can share many more wonderful moments together.

A special thank you goes to Silvia, who, with so much generosity and love, has always supported me and forgiven me for all the time taken away by my studies.<3

I thank my parents, who made all of this (and so much more) possible, always encouraging me to give my best and to never set limits for myself.

Lastly, but not least, I thank myself for making a lots sacrifices and for always believing in it.

Thanks!

Inizio ringraziando vivamente il Prof. Silvio Ranise per avermi dato la possibilità di scrivere questa tesi. Sono immensamente grato per aver potuto intraprendere il periodo di tirocinio e scrittura della tesi in un ambiente così prestigioso come la Fondazione Bruno Kessler (FBK), ambiente che mi ha permesso di conoscere nuove realtà e di crearmi un bagaglio tecnico impagabile. Particolari ringraziamenti sono dedicati al Dr. Stefano Berlato e al Dr. Riccardo Longo che mi hanno accompagnato in questo magnifico percorso dandomi sostegno dall'inizio alla fine. Grazie alla loro esperienza, saggezza e dedizione sono riuscito ad esplorare nuovi argomenti in un settore, nel quale la mia conoscenza era pressochè basilare, fino al punto di apprenderli e applicarli nella mia tesi. Ancora nessuno scienziato è riuscito a calcolare la quantità di pazienza che avete avuto con me e per questo ve ne sono infinitamente grato e vi auguro il meglio per il futuro. Un ringraziamento anche all'Università di Trento, al dipartimento DISI e a tutti i professori che mi hanno permesso di raggiungere questo traguardo e di poter ottenere questa laurea.

In seguito, voglio ringraziare mio fratello Carlo per aver reso possibile il mio percorso universitario dandomi l'aiuto necessario e insegmandomi le, purtroppo assenti, basi (matematiche) per riuscir ad affrontare i primi esami. Senza di te penso che sarebbe stato alquanto difficile concludere questo percorso. Grazie.

Poi voglio ringraziare gli amici di una vita: Sepp, Pie, Cristl, Mena, Pippo e per ultime Ari e Uoua :), che ci siete sempre stati per qualsiasi cosa (anche per dare fastidio); i migliori coinquilini del mondo: Slanderlor (anche se non fa niente) e chef Pippo; poi Zano, con il quale ho trascorso assieme 14 anni dietro ai banchi e che senza di te questi 3 anni universitari sarebbero stati vuoti e anche i compagni di studiate in BUC: Bertol e Zanet. Ci sarebbe una lista infinita di amici, parenti e persone che rispetto e che vorrei elencare, ma purtroppo l'inchiostro costa e chi mi conosce capisce. Comunque, voglio ringraziare tutti voi che siete in questa lista e spero vivamente di condividere ancora tanti bei momenti assieme.

Un ringraziamento speciale è dedicato a Silvia che con tanta generosità e amore mi ha sempre supportato e perdonato per tutto il tempo sottratto dallo studio.<3

Ringrazio i miei genitori che hanno permesso tutto questo (e tanto altro), incitandomi sempre a dare il massimo e a non pormi mai limiti.

Per ultimo, ma non d'importanza, ringrazio me stesso per aver fatto tanti sacrifici e per crederci sempre.

Grazie!

Contents

Glossary

ABAC Attribute-Based Access Control

AC Access Control

ACL Access Control List

AWS Amazon Web Services

CA Certificate Authority

CAC Cryptographic Access Control

CRUD Create, Read, Update and Delete

CSP Cloud Service Provider

DAC Discriminatory Access Control

HF Hyperledger Fabric

IK Intermediary Key

IP Intermediary Proxy

MA-ABE Multi-Authority Attribute Based Encryption

MAC Mandatory Access Control

MM Metadata Manager

MNO Mobile Network Operator

PDC Private Data Collection

PRE Proxy Re-Encryption

RBAC Role-Based Access Control

RM Reference Monitor

Abstract

With the development and widespread adoption of the Cloud, data sharing among individuals and organizations has become more convenient than ever before. However, one of the main obstacles that hinders the adoption of Cloud concerns data security and, in particular, data confidentiality and integrity, such as the risk of unauthorized access by third parties, data breaches due to vulnerabilities in the Cloud Service Provider (CSP)'s infrastructure, lack of control over data location, internal attackers and malicious users. Additionally, CSP are often considered *"honest but curious"*, meaning they might have the motivation or the technical ability to analyze and access to the data they store or process.

The arrival of blockchain technology brought a new secure and decentralized approach ideal to guarantee the confidentiality and the integrity of sensitive data, especially in cross-organizational scenarios where multiple equally (dis)trusting parties need to collaborate and exchange data securely. Nonetheless, the blockchain is a difficulty technology to set-up, maintain, and use. For this reason, the adoption of blockchain technology in a serverless fashion is sometimes preferred, in which the duty to manage the blockchain is delegated to a *partially-trusted* third party, e.g. usually a CSP.

The fact that organizations use Cloud-managed blockchains poses again a risk to the confidentiality of data, as the CSP — which handles the nodes composing the blockchain — has ideally access to the data contained within the transactions.

In this thesis, we propose a solution for guaranteeing the confidentiality of sensitive data shared over a Cloud-managed blockchain network managed by a partially trusted CSP in cross-organizational scenarios. The solution ensures the confidentiality of the shared data by combining Cryptographic Access Control (CAC) — for end-to-end protection of the data according to Access Control (AC) policies internal to organizations — with Proxy Re-Encryption (PRE) — for achieving effective and secure revocation of access privileges. This project follows a deep analysis of a previous solution proposed by Enrico Marconi [11], which introduces a multi-admin data management system designed to ensure the confidentiality of sensistive data shared on a blockchain network managed by an honest-but-curious CSP. We identify its limitations, such as problems with collusions and the lack of a revocation mechanism, and outline the requirements that our solution must address. Finally, we implements a proof-of-concept to demonstrate the feasibility of the solution.

1 Introduction

Context There are several scenarios in which different organizations need to share sensitive data among themselves. For example, consider the sharing of electronic medical records between hospitals and clinics, or data exchanges between different branches of a company. Often, as in the case of medical records, the level of importance and the need for data confidentiality is extremely high. Designing a solution for secure data sharing in cross-organizational scenarios has long been a significant challenge. In particular, applications handling such highly sensitive data must ensure its confidentiality by preventing unauthorized accesses through the use of cryptography and effective AC mechanisms.

Problem For years, Cloud platforms were deemed a natural fit for multi-organizational data management systems. However, even though Cloud is a cost-effective and convenient solution for data sharing, it lacks an important ingredient; it does not give organizations full control over their own data. To date, there have been many incidents of privacy invasions, as users do not placed the right trust in Cloud[1][2][3][4].

In 2008, with the publication of [12], blockchain technology emerged, offering new possibilities for the future of web applications. Blockchain-based systems utilize a decentralized approach, where power and control are distributed among all participants rather than being concentrated in a single entity. This technology aims to distribute trust by ensuring transparency through an immutable public ledger. The design of the first blockchain proposed in [12] featured a public, permissionless system with pseudonymous actors, without addressing confidentiality concerns. Recently, as blockchain technology has evolved to accommodate broader use cases, confidentiality has become a key focus, leading to the development of permissioned blockchains. Unlike permissionless blockchains, permissioned blockchains require all participants to authenticate and obtain authorization from current members before joining the blockchain. This makes permissioned blockchains a more suitable choice for multi-organizational data management systems. However, while permissioned blockchains are well-suited for these applications, they are complex to set up, maintain, and use. Consequently, many organizations are increasingly opting to delegate this challenging task to CSPs such as AWS[5], using a serverless approach. Despite their numerous advantages, such solutions face significant challenges. In fact, CSPs, while generally trustworthy, are considered "honest-but-curious" when it comes to sensitive data. Additionally, a blockchain managed by a third party is vulnerable to both internal and external threats.

Solution and contributions In this thesis, we propose a solution for guaranteeing the confidentiality of sensitive data exchanged over a Cloud-managed blockchain-based application. The solution combines CAC and PRE approach with blockchain smart-contracts to enforce AC policies on data exchanged on a permissioned blockchain managed by a partially trusted CSP. As result, our solution allows its users to securely share sensitive data among themselves, while retaining sovereignty over their data, in a completely transparent and auditable manner. In particular, this thesis makes the following contributions:

- we analyze the solution proposed by Enrico Marconi[11], which introduces a multi-admin data management system designed to ensure the confidentiality of sensistive data shared on a blockchain network managed by an honest-but-curious CSP. We identify its limitations, and outline the requirements that our solution must address.

1 `https://www.wired.com/story/voter-records-exposed-database/`
2 `https://www.wired.com/story/amazon-s3-data-exposure/`
3 `https://www.wired.com/story/exactis-data-leak-fallout/`
4 `https://www.wired.com/story/security-news-this-week-the-pentagon-left-data-exposed-in-the-cloud/`
5 `https://aws.amazon.com/`

- we propose a novel solution that addresses the limitations previously identified;
- we implement our solution in a proof-of-concept to demonstrate the feasibility of our solution.

The thesis is structured as follows: in Chapter 2 we analyze state-of-art solutions that share some kind of similarities with our work, comparing them with our approach. Then, in Chapter 3 we describe some basic knowledge for the main aspects and technologies treated in this thesis; afterwards, in Chapter 4 we first describe the reference scenario for our work, then we give an overview of our solution's design and after we describe each components of the solution. Then, we provide an in-depth description of the cryptographic scheme presented in the solution and finally we describe the main operational flows of the solution. In Chapter 5 we discuss the implementation of the proof-of-concept of the solution; Finally, in Chaper 6 we evaluate the work, summarize our findings in the conclusion, and outline directions for future works.

2 Related Work

All started with the pubblication of [12] in 2008, where a new technology called *Blockchain* was introducted. This technology was at first used for an eletronic payment system, but afterwards it gained a lot of attention and researchers from all over the world started to develop new potential applications of such technology in fields where the confidentiality of the data is of significant importance.

Leveraging the potential of the blockchain, the authors of [16] implemented a protocol that allows the users own and control their data granting an high level of privacy. The blockchain is turned into an automated access-control manager accepting from the user and service providers two types of transactions: one that allows the user to manage the access-policies of a certain service provider, one that allows the user or the service provider to storage or retrieve the data. In this project, there is a limitation in which the access-policies are defined as the type of data that the provider can access and in case the provider is not fair, it can label the data in a different way and access to unauthorized data. In our solution, we cannot give any amount of trust to the entities interacting with the system, instead we must treat every entity as honest-but-curious. Therefore we avoid the possibility of an entity to access or manage unauthorized data through a further CAC enforcement mechanism that obfuscates the data for all the entities in the system, so that the entities manage only encrypted data.

Also in [13] the authors implement a solution where the blockchain is used as access control system, in particular they develop an identity-based and role-based access control system with blockchain. The scenario is different from ours, the use case applies only to an organization, that owns all the data and manages its own users. Our scenario instead involves many organizations that shares data among them.

In [5], as in our solution, the authors propose a solution in a multi-admin scenario, where the access control system is enforced through a combination of cryptographic techniques, called Multi-Authority Attribute Based Encryption (MA-ABE), and smart contracts. The limitations of this approach is the shortage of libraries that support MA-ABE, for example OpenABE[1] and Mosaic[2] which have not been updated in recent years and not been formally proven to be secure. To address this limitation, in our solution we propose a PRE scheme, described in [8], which also enable our system to be cryptographic-agnostic: every organization can enforce AC with whatever AC mechanism they desire, as long as the re-encryption of data is carried out with keys from the PRE.

In [7] the authors describes a solution that permits several Mobile Network Operator (MNO) of different countries to share data in a secure and dynamic way. The system is based on a permissioned blockchain and allows the MNOs to share data either publicly, with all the partecipants of the network, or privately, to a subset of MNOs. The implementation is in Hyperledger Fabric (HF) making use of its Private Data Collection (PDC) mechanism. For every MNO there are three roles: MNO Admin, MNO Users and MNO Guests; since this group of actors has the same set of permissions of the peer itself, this means that only one peer of the network must exist for every MNO. In our solution each organization has the possibility to share data through multiple peers, but at least one, this is possible through the presence of Reference Monitor (RM) that acts as the interface component between the organizations's users and the blockchain network. The confidentiality of the data in [7] is relied to the blockchain and PDCs that are implemented on the cloud. In our solution the confidentiality of the data is renforced through a CAC mechanism that allows only the authorized recipient to have access to a certain resource and making every resource unaccessible by the partially trusted entities, e.g. blockchain network. The use of PRE has a crucial importance in our solution. It is first appearance happens in [14] and then it has be broadly used for many data sharing applications, among which it appears also in [10] and [2] where PRE is combined with the blockchain. Both [10] and [2] solutions

1 `https://github.com/zeutro/openabe`

2 `https://github.com/marcellop71/mosaic`

involved blockchain and PRE to build a mechanism that allows to safely stores data by IoT devices on the cloud and allowing other users to access the data through PRE. Altough these solutions propose a similar approach to ours, they differ for the trust level they assume on the entities of the system. In particular, in [10], the blockchain acts as a trusted entities that initialize the system's secret parameters as well as the users's parameters. In our solution, we do not give any level of trust to the blockchain that acts as an untrusted entity.

Another important aspect is the storing of the data on the Cloud. From this arises the need to enforce an AC mechanism on these data, while at the same time providing a high level of confidentiality in the cloud. In [4], Garrison et al. develop a CAC scheme to broker access to outsourced data through Role-Based Access Control (RBAC) and Hierarchical Role Based Access Control ($RBAC_1$) policies using either Identity-Based Encryption (IBE) or Public Key Encryption (PKE). This research points out how this technique introduces an overhead if implemented in a heavily dynamic scenario, with many revocation and re-keying procedures. In our solution we managed to partially solve these issues through the use of PRE, which allows us to store data and re-encrypt as needed, instead of having to store multiple copies of the same data fragment, where each copy is meant for a different owner and thus for a different decryption key.

The researcher in [9] propose a system based on PRE, which is used to enforce fine-grained AC policies on data stored on a honest but curious networked storage system. In a similar fashion to what we propose, AC policies are cryptographically enforced: data are encrypted with the owner's key, whereas access to other users is granted through a delegation key, which enables them to re-encrypt the data under their own key. In an analogous way, in [15] the research enforces fine-grained AC policies based on data's attributes, where these data are stored on cloud. The system they propose is again based on PRE, but it is also combined with Attribute-Based Encryption (ABE) and lazy encryption.

It is important to mention that our solution comes out as a continuation of a previous project conducted by Enrico Marconi [11], which introduces a multi-admin data management system designed to ensure the confidentiality of sensistive data shared on a blockchain network managed by an honest-but-curious CSP. After having analyzed the project of Marconi, we found some limitations in the architecture. In particular, there are some problems with permissions, collusions, authorizations and revocations. In fact, in his solution the RM, as component responsible for performing the PRE, is in possession of the re-encryption keys that transforms a resource into another resource, where the first resource could only be decrypted by the intended organization's private key, that is in possession only by the admin of the organization, but after the re-encryption step the resource can be decrypted by the private key of the authorized role. Thereby this gives to the RM the possibility to make use of a re-encryption key in his possess for an unauthorized role. We remember that the RM is managed by a cloud entity and so it is a untrusted component. Another problem is during the phase in which happens the encryption of the resource. Knowing that the user who creates a resource can do whatever they want with it, we want to emphatized the fact that in the solution of Marconi is the user the one who decides with which organization's public key encrypting the simmetric key. This gives to the single user the possibility to share the key with unauthorized organizations. We address this issue by creating a pair of key for each PDC with which the user can encrypt the resource she wants to share and where the action by the user on the PDC is controlled by the AC policies. In term of collusions, Marconi resolves this problems giving full authority to RM that controls accesses with re-encryption keys. RM, as partially trusted component, should not have the full authority to controll accesses with key in his possess; to address this problem we introduce in the system the fact that every shared data belongs to a PDC and that every PDC has a private-public key pair. The problem related with the revocation of an user is addressed introducing the Intermediary Proxy (IP) and the Intermediary Key (IK). In fact, in case of a revocated users the system has to change only the IK and the related re-encryption keys and not all the organization's key pair, that would incurs significant computational overhead. To sum up, in our solution we aim to give the RM fewer privileges, seeking to limit the trust placed in it as much as possible.

3 Background

In this chapter, we introduce the basic knowledge for the most important concepts present in the thesis. In Section 3.1 we explain the concept of blockchain, its components and the different forms it takes. After, we introduce Hyperledger Fabric and its characteristics. Then, in Section 3.2, we introduce the concept of hybrid encryption, including also a graphical representation, while in Section 3.3 we descripe the proxy re-encryption approach integrating an example to better understand how it works. Finally, in Section 3.4.1, we introduce the features of an access control mechanism by listing some of the most common models and then provide a detailed explanation of CAC.

3.1 Blockchain

Cryptography underlies several important data structures including the blockchain, a specific instance of a distributed ledger. A distributed ledger is a database replicated across multiple nodes (e.g., computing devices) connected to the same peer-to-peer network where the database is updated through consensus mechanisms to ensure consistency (i.e., the local databases on each node contain the same data). The blockchain is an instance of distributed ledger where the data (contained in so-called transactions) exchanged between nodes (called peers) are organized in a chain of blocks that allows the addition of new blocks but not the modification of existing blocks . Each block is cryptographically linked to the previous block, providing high resistance to tampering. The blockchain makes extensive use of asymmetric cryptography, particularly for creating digital signatures on transactions. The way in which a node can access a network and participate in a blockchain identifies the following types of blockchain:

- **public permissionless blockchain**: without a central authority, any node can participate in a public permissionless blockchain, often without prior authentication. This type of blockchain often includes economic incentives for participating nodes but is usually characterized by limited scalability. Some of the most well-known blockchains (e.g., Bitcoin[1], Ethereum[2]) fall under this category;

- **private permissioned blockchain**: with a central authority, only authorized nodes can participate in a private permissioned blockchain, usually after the authentication of the organization or user managing the nodes. This type of blockchains provide a controlled, secure, and private environment. ;

  - **consortium-based blockchain**: the central authority managing a private permissioned blockchain can correspond to a consortium of organizations. This type of blockchain is usually used in an enterprise context, where multiple organizations collaborate and exchange (often sensitive) data;

- **hybrid blockchain**: this type of blockchain blends characteristics of public permissionless blockchains and private permissioned blockchains.

A blockchain can incorporate **smart contracts**, which are distributed applications (i.e., software) executed by nodes participating in the blockchain. All nodes must derive the same result from executing a smart contract, and the result is then usually recorded within the blockchain as a transaction.

1 `https://bitcoin.org`
2 `https://ethereum.org`

3.1.1 The Hyperledger Foundation and Hyperledger Fabric

The Hyperledger Foundation[3] is a consortium supporting an ecosystem of open-source software projects for developing blockchain-based applications, such as HF[4]. HF provides blockchain services with a modular approach that allows users to choose various components (e.g., consensus mechanism, identity management, and smart contract handling). The main characteristics of HF include:

- **channels**: in HF, a channel corresponds to a blockchain. In other words, HF allows for multiple independent blockchains, each with potentially different subsets of participating nodes;
- **PDC** : in HF, a channel can have one or more PDCs. A PDC is a subset of data that is shared only among a specific subset of nodes in a channel. PDCs are useful when two or more nodes participating in a channel want to exchange data in a way that hides the content of transactions (i.e., the data) from other nodes in the channel, while still letting all participants know that a transaction has occurred. In a PDC, data is shared only among the subset of nodes participating in the PDC, while a hash of the data is shared with all nodes participating in the channel as part of the transaction. The AC of a PDC is controlled by chaincodes and PDCs configuration ensuring that the data are only accessible to authorized nodes within the channel;
- **chaincodes**: a chaincode is a deployment package for one or more smart contracts. Typically, a chaincode contains a single smart contract. In HF, smart contracts[5] can be written in Go, Node.js, or Java ;
- **identity and access**: in HF, blockchain management includes consortium-based setups. Within HF, digital identities are encapsulated in X.509 digital certificates[6] issued by a Certificate Authority (CA) component specific to HF. Each partecipant in the network has a unique identity that is cryptographically secure and verifiable, while AC policies can be expressed using Access Control List (ACL) based on the Attribute-Based Access Control (ABAC) model.

3.2 Hybrid Encryption

Hybrid encryption consists in the combined use of symmetric and asymmetric cryptographic algorithms. Both kinds of algorithms expect the creation of cryptographic keys, symmetric algorithms use k^{sym} for both encryption and decryption of data, while asymmetric algorithms use sk and pk for encryption or decryption of the data. Hybrid encryption exploits the advantages of both kinds of algorithms in term of usability, efficiency, and performance. In fact, the asymmetric algorithm having a key-pair, one secret and one public, eliminates the need to securely exchange the unique secret key between two parties avoiding the risk of being intercepted by an unauthorized party. While the symmetric algorithm is well-known for its speed and requires less computational power, making it suitable for encrypting large amount of data. We can see a graphical representation of hybrid encryption in Figure 3.1. k^{sym} is generated for encrypting a plaintext m, obtaining a ciphertext c. Then, k^{sym} itself is encrypted using sk, so that only the possessor of pk can decrypt k^{sym} and consequently access to m.

3.3 Proxy Re-Encryption

PRE is a type of asymmetric cryptographic algorithm where a semi-trusted *proxy* alters a ciphertext meant to be decrypted by a key so that it can be decrypted by another key . The main concept is that the *proxy* never accesses m. We can see a graphical representation of PRE in Figure 3.2:

- *Alice* encrypts m using its public key pk_{Alice} obtaining the ciphertext c;
- *Alice* sends c to the *proxy*; at this moment, c can be decrypted only with sk_{Alice};

3 `https://hyperledger.github.io/`
4 `https://www.hyperledger.org/`
5 `https://hyperledger-fabric.readthedocs.io/en/release-2.5/smartcontract/smartcontract.html`
6 `https://datatracker.ietf.org/doc/html/rfc5280`

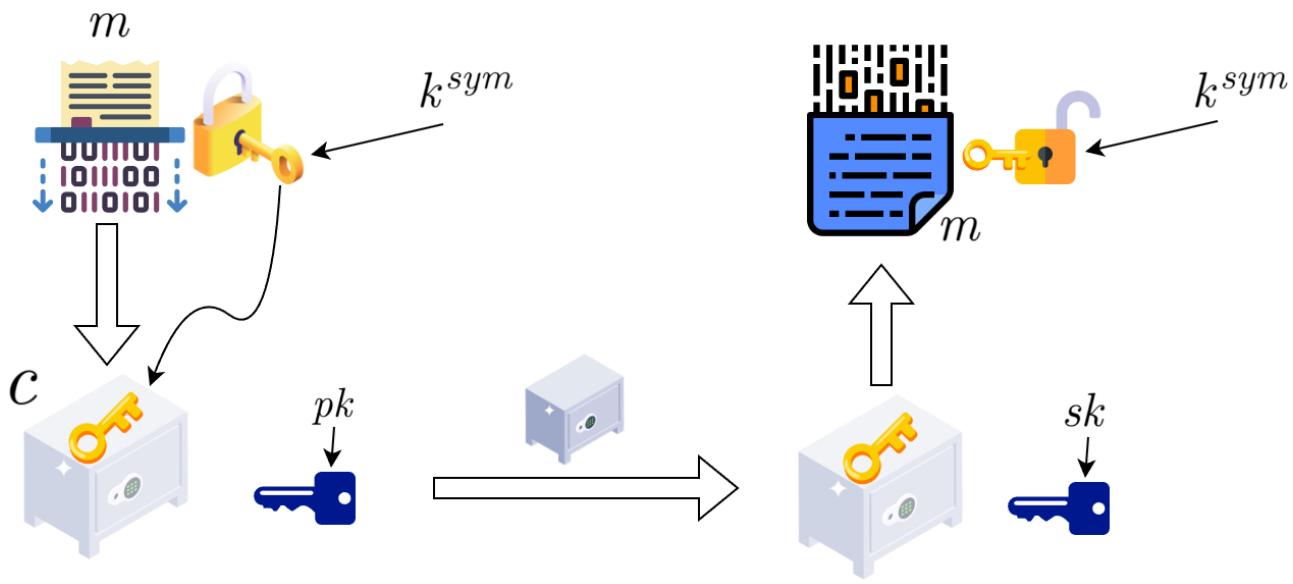

Figure 3.1: Hybrid Encryption

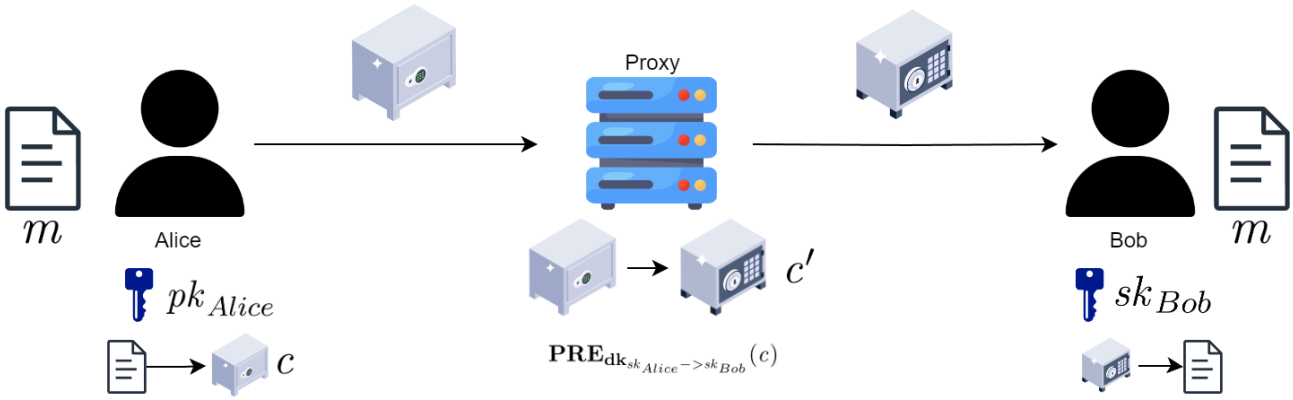

Figure 3.2: Proxy Re-encryption

- if *Alice* wants to give *Bob* access to c , she creates a delegation key $\mathbf{dk}_{sk_{Alice} \to sk_{Bob}}$ and sends it to the *proxy*;
- *proxy* computes the proxy re-encryption $\mathbf{PRE}_{\mathbf{dk}_{sk_{Alice} \to sk_{Bob}}}(c)$ obtaining c';
- *Bob* decrypts c' with sk_{Bob}, obtaining the plaintext m;

In the above example we consider the use of PRE with asymmetric encryption, but it is important to note that the PRE can be combined also with the hybrid encryption, explained in Section 3.2, exploiting the cited advantages of it.

3.4 Access Control

AC is a very important aspect of computer security and it is present in every computer system. We can describe AC as the process that decides whether a user of a system can have access to a determined resource. AC can be instantiated in different models, the most common models being:

- Discriminatory Access Control (DAC): the resource owner decides who can access to the resource and with what permissions;
- Mandatory Access Control (MAC): an administrator assigns a security label to each resource, and users can access resources based on their security clearance;

- ABAC: access to resources is granted after checking a policy which is defined using attributes describing characteristics of the involved user, resource, and context in which the access request occurs.
- RBAC: users are assigned to one or more roles and roles are assigned to one or more permissions. The state of a RBAC policy can be described as a tuple < **U**, **R**, **F**, **UR**, r_n>, where **U** is the set of users, **R** is the set of roles, **F** is a set of resources, **UR** is the set of user-role assignments and r_n is a set of role-permission assignments;

3.4.1 Cryptographic Access Control

In order to enforce AC mechanism, CAC is a mechanism that uses cryptographic methods to ensure that only authorized users can access the resources, adding an extra layer of security. CAC can be implemented in centralized or decentralized AC enforcement mechanisms; in the first one all AC decisions and policies are managed and enforced by a single, central authority, while in the latter one AC decisions and policies are distributed across multiple authorities, that can set its own AC policies, being extremly useful in decentralize applications (such as the blockchain). At a high level, CAC secures data by encrypting them with cryptographic keys, so that only the users in possession of the correct keys are able to decrypt the data and thus the data are opaque to everyone but the authorized recipients. In systems where AC policies are enforced by partially trusted entities, e.g., a cloud provider , CAC can be the solution that reduces the threat posed by the partially trusted entity.

RBAC and CAC CAC can be used to enforce RBAC policies. In this mechanism, every role is equipped with a public-private key pair for encryption and decryption. Only users assigned to that role are in possess of the private key and so be able to decrypt encrypted data intended to be shared with that role.

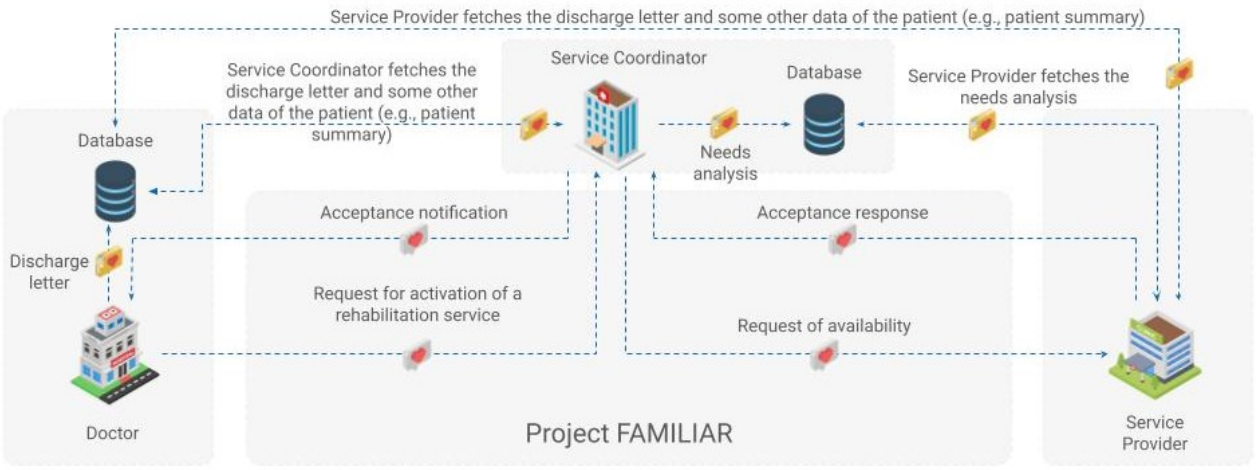

Figure 4.1: Use case of the project FAMILIAR

4 Solution Overview

In this chapter, we provide the reader, in Section 4.1 the description of the scenario under consideration, including a representative use case, followed by the requirements that the solution addresses. In Section 4.2, we give an overview of the architecture of the solution describing the technlogies and paradigms used and the components present. While in Section 4.3, we provide the description of the cryptographic scheme of the solution with a graphical representation of all steps. Finally, in Section 4.4, we explain some of the basic operations of the solution.

4.1 Reference Scenario and Motivations

Our scenario involves a consortium of organizations that need to exchange data securely and confidentially among themselves. More in detail, we consider a current research and innovation project — called FAMILIAR[1] ("piattaForma dinAMica per IL socIosanitARio") — that is being carried out in the Trentino region in Italy. FAMILIAR aims at improving the collaboration in the healthcare sector (in particular the management of data exchange) through the development of a platform that enables the creation of a coordination system for Long Term Care (LTC) services using innovative digital technologies. The project is focused on the field of dementia and the goal is to increase the quality of life for patients and support for their families. .

A Representative Use Case As an example, we consider the use case presented in FAMILIAR where it is considered the management of a patient with mild cognitive impairment that requires cardiac rehabilitation after the discharge from the hospital following a surgical procedure (e.g. the implantation of a pacemaker). This use case involves the following entities: a **doctor**, a **service coordinator** and one or more **service providers**.

The process begins with the sending of the **discharge letter**, the **request for activation of a rehabilitation service** and some other data of the patient (**patient summary**) by the doctor to the service coordinator. The service coordinator visualizes and analyzes the data creating a **needs analysis**. Finally the service coordinator individualizes one or more service providers sending them the **request of availability** and the needs analysis. The service providers visualizes the data sendt by the service coordinator and retrieves the data of the patient (discharge letter and patient summary).

1 `https://aleph.fbk.eu/projects/FAMILIAR`

The service provider checks its availability and sends back to the service coordinator an **acceptance response**. The service coordinator visualizes every service provider's acceptance responses and informs the doctor through an **acceptance notification**. At the end the doctor gets in touch with the available service providers and the patient to manage and coordinate the collaboration.

In the example we observe the presence of the following organizations: hospital, service coordinator and one or more service providers. For the data exchange between these organizations we denote a single channel (**patientManagement**). The organizations share the data on different PDCs depending on the types of data, we denote the PDCs in the Table 4.1.

We assume that every organization can have its own internal structures (e.g. RBAC or ABAC) to distinguish the type of data recipient and to better control data access. For this example, we chose RBAC as model to manage permissions and accesses for every organization. We identify the following roles: doctor (hospital), staff member (service coordinator) and staff member (service provider). Each organization decides the roles that can access the PDCs. The policies of the organization are structured like this: `<Role, Action, PDC, dataType>`, where `Action` is one among those of the Create, Read, Update and Delete (CRUD) paradigm. An example of a policy could be:

```
<Doctor, CREATE, PDCPatientInformation, [demographics, medicalHistory]>
```
(4.1)

Requirements In the scenario, we operates in a inhomogeneous context where the organizations handle sensitive data and where the security of these data is crucial. In our solution, we focus on these requirements:

- **use of cloud services (R1)**: the system has to operates on the cloud taking advantages that it offers such scalability, availability and flexibility;
- **use of permissioned blockchain (R2)**: the system must include the use of a permissioned blockchain as data manager taking advantages that it offers such tracebility, authentication, integrity, non-repudiability;
- **confidentiality (R3)**: every data shared in the system has to be encrypted, only the authorized recipients have access to the plain text, for every other entities in the system the data are obfuscated and cannot be decrypted;
- **access control (R4)**: data must only be accessible to authorized recipients, preventing any collusion between users and partially trusted entities.
- **access control methods (R5)**: it is important that each organization is able to implement its own access control method, e.g. RBAC or ABAC;
- **revocation (R6)**: organizations must be able to securely revoke their users from one or more roles, ensuring that the revoked users no longer have access to the resources associated with those roles.

Table 4.1: **PDCs Table**

PDC name	Partecipanting Organizations	Data shared
PDCPatientInformation	Hospital - Service Coordinator - Service Provider	Patient summary (demographics, medicalHistory), discharge letter
PDCNeedsAnalysis	Service Coordinator - Service Provider	Needs analysis
PDCTreatmentInformation	Hospital - Service Coordinator - (Patient)	Data about patient treatment

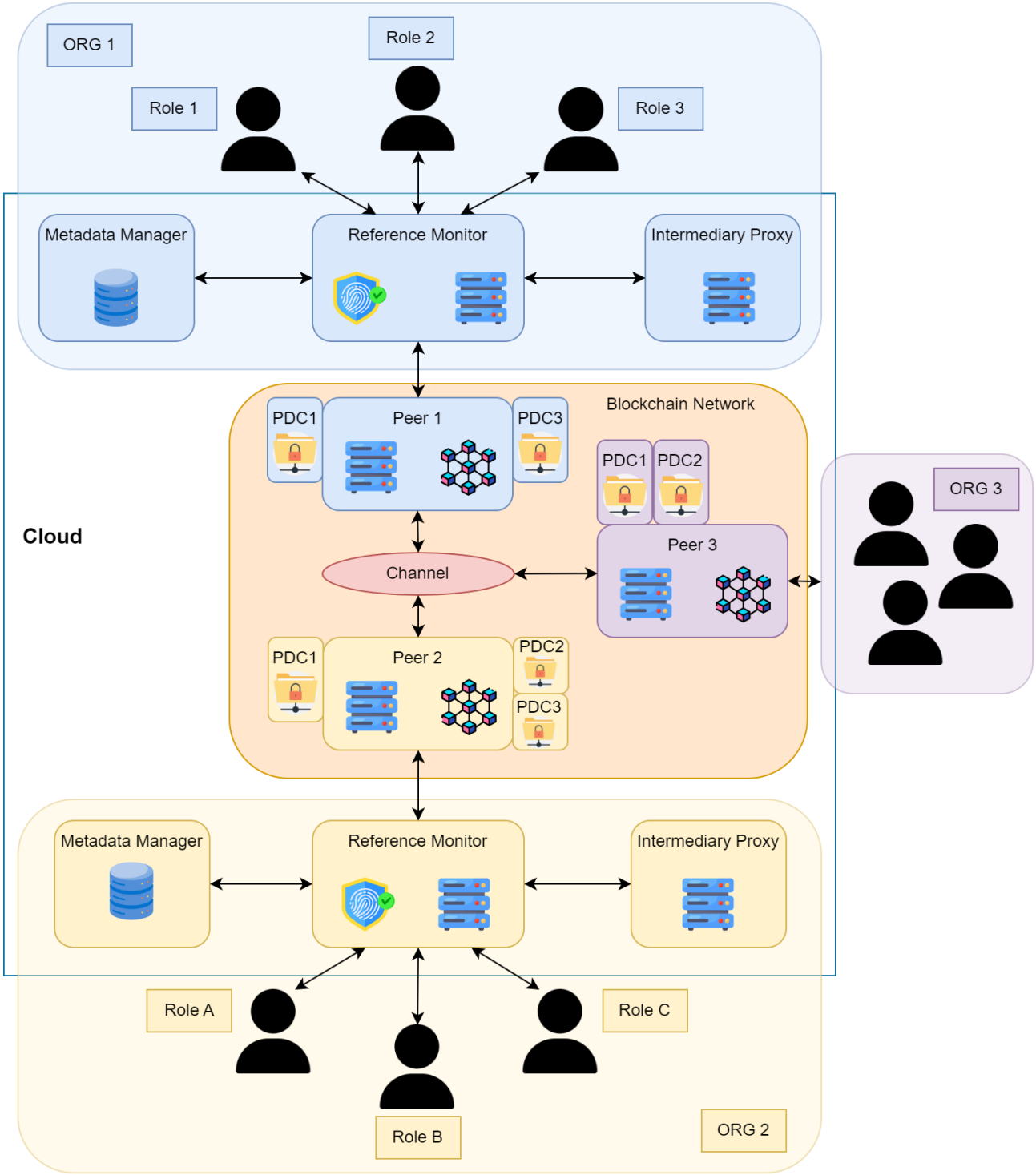

Figure 4.2: General architecture of the project

4.2 Architecture Overview

In this section, we present the architecture overview of the solution, in Section 4.2.1 we introduce the technologies and paradigms present in the system explaining the motivation with reference to the requirements in the previous section. In Section 4.2.2, for every component present in the system, we explain its tasks and functions.

4.2.1 Technologies and Paradigms

The *cloud* is nowadays a very useful and promising technology . Its peculiarities such as accessibility, scalability and cost saving are the reason why this technology has been chosen by the project FAMILIAR. In fact, the technology solution presented in FAMILIAR consists in a cloud-managed private

permissioned blockchain. In particular, it uses HF through Amazon Managed Blockchain of Amazon Web Services (AWS) [2] so to reduce the setup and configuration effort of organizations and inherit the peculiarities of cloud computing (accessibility, scalability). However, while cloud providers are usually trusted for guaranteeing the integrity and availability of data and services, the same cannot be said for the confidentiality of the data. In fact, cloud providers are usually assumed in the literature to be "honest but curious" [4] [6] [3]. Therefore, in our solution, all data transmitted to the cloud are previously encrypted and rendered inaccessible by the CSP providing a high level of *confidentiality*. Every organization must join the blockchain network with at least one peer node, more node can be used to achieve redundancy and higher throughput. The peers are responsible for exchanging and storing data for their organization, by actively interacting with the blockchain. The peers contain the smart-contracts that are invoked by the organization in order to access to data of the network. For every action turns on the data of the network, a transaction is pinned down on the channel's ledger. The channel ledger is the record of all transactions and states specific to a particular channel within a blockchain network. It serves as the authoritative source of truth for all the operations and state transitions that have occurred on that channel. In our solution, the organizations share data on PDCs; transactions made on PDCs are visible in an encrypted manner in the shared ledger of the channel, so it can be noticed that a transaction has been made but its content cannot be seen, except by the participating organizations. When a PDC is configured, the participating organizations are included without specifying internal roles. Each organization decides internally how to manage the PDCs's data. Transaction are created by executing chaincodes residing on the peers. For our solution, we propose in every chaincode of the PDCs the possibility to execute each CRUD action. Since Read action does not modify data there is no need in the chaincode to create a transaction and to change the channel ledger. The chaincode is, by the way, crucial to apply the AC policies on the data. HF uses ACL to control accesses to blockchain data and invoke smart-contracts. In our reference scenario, the use of ACLs does not fulfill the level of security, as the cloud provider that manages the blockchain could easily tamper with ACLs to let a malicious user access whatever data he/she desires. To preserve the confidentiality and a better access control of blockchain-hosted data in presence of a honest but curious cloud provider, we consider the use of CAC (Section 3.4.1). As AC method of the organizations, we decide to take into consideration in our solution only RBAC. RBAC descriminates AC policies based on user's role. It is important to emphasize the aspect in which each organization can define in general its *own AC method* and in our system using RBAC every organization can define as many roles as it needs without other organizations within the system being aware of them. Combining CAC and RBAC our solution improve the level of security ensuring that only the authorized recipient can have access to the resources. In the solution, we consider to perform the PRE (Section 3.3) on the resources to permit that every organization can decide to which of its users/roles the resource is addressed. In [11] the resource is encrypted with the public's key of the recipient organizations and later it is performed the PRE on the resource to allow that the resource can be accessed by the designed user/role. The PRE is performed by a partially-entrusted entity and we found that using the approach designed by Marconi it may cause problems of collusion. In other words, the proxy that performs the PRE can be corrupted and can perform the PRE for unauthorized users. Therfore, we introduce in the system the fact that every shared data belongs to a PDC and that every PDC has a private-public key pair. These keys are shared only with the administrator of the authorized organizations. The resources are encrypted with the PDC's key and only through the PRE the resource can be accessed by the user/role. This concept is better explained in the Section 4.3. Introducing the key pair for every PDC we resolve the problem of collusion because the partially-trusted cannot perform the PRE for users/roles that do not belongs to the PDC. However, using CAC introduces a key management problem: *revocation*. In other words, it is the situation when a user loses their permissions and they are no more authorized to access to determined data. To resolve this problem we introduce a new component in the solution: the IP. The concept of revocation and its issues is better explained in the Section 4.4.4.

2 `https://aws.amazon.com`

4.2.2 Components Description

In this section, we introduce the main components of the solution. Every component is crucial in order to ensure the required level of security and to meet the imposed requirements.

Reference Monitor. The RM is a central component of our solution, it is a vital component for the AC system, since it is the manager of the AC policies for every access request. We assume the RM to be managed by the CSP, which is an "Honest but Curious" entity, thus we cannot give the RM full trust, which is anyway already been taken care by cryptographic construction of the solution. Every organizations has its own RM, acting as interface component between the organization's partecipants and peers of the network. The fact that every organization has its own RM allows to freely handle their internal AC using whichever AC mechanism they want, and this increments the flexibility of our solution.

The tasks of the RM are the following:

- **Authentication**: in order to interact with the system the user must first authenticate themself to the RM using the certificate of the blockchain network and the signing key-pair;
- **RM's ACL**: the user that want to interface with the blockchain network send the request to the RM, which parses the request and control through the ACL if the user's request is allowed. The ACL is provided by the administrator and enforce fine-grained AC on the data handled by the system. Policies are defined as `<rolez, action, PDC*, dataType>`, where `role` is any of the roles defined by the admin, `action` can be any of CRUD, `PDC*` can be any of the PDCs in which the organization partecipates and `dataType` are a more optimized specification of shared data types shared in the PDC for a better and more accurate management;
- **Interaction with Blockchain Network**: after having authenticated the user and checked the request with the policies, the RM interacts with the peer node of the blockchain network in order to fulfill the request of the user, invocating the appropriate chaincode;
- **Proxy re-encryption**: another major task of the RM is to carry out the re-encryption step by acting as a proxy. The RM fetches the resource from the blockchain network and performs the first re-encryption of the encrypted symmetric key present in the asset. The re-encrypted key is then sended to the IP for the second re-encryption.

Intermediary Proxy. The IP is another major component of the architecture. It performs the second PRE that makes the symmetric key, which was be made encryptable only by the IK's private key, decryptable by the authorized role. With the introduction of the IP, we have to recall the presence of the IK. The IK consists in an asymmetric key pair and its fundamental for the creation of the re-encryption keys assigned to the RM and IP which allows the two PRE steps. To every role is assigned one IK. IP and IK enable the solutions of some problems like revocation (Section 4.4.4) and collusion.

Metadata Manager. The Metadata Manager (MM) is a component of the system that allows the RM to store information in order to share data with the network. The MM only communicates with the RM, who is responsible for inserting the required data, updating and retrieving it as needed by its users. The content of the MM may be extended to accommodate for future use cases, but at its minimum, it must contain all necessary information that allows the RM to authenticate and authorize the organization's users.

Admin. The Admin is a component of the application, every organization must have at least one Admin. The Admin is responsible for the internal management of the organization. The tasks it carries out are:

- **Managing roles**: the Admin has the task to establish the roles of the organization, to create the cryptographic keys and IK for every role and to send them to the users of that roles.

- **Managing PDCs**: the Admin is the only owner of the private keys of the PDCs in which the organization is partecipating and of the private key of the organization, fundamental for the sharing of the PDCs keys. When the organization partecipates to a new PDC, the Admin has the task to decide which roles can execute requests on that PDC.
- **Managing Policies**: the Admin creates the policies deciding which role can execute which request on what PDC and dataType and then he has the authority to add, modify or delete the policies.
- **Revocation**: in case of revocation, i.e. a user not belonging to a role any more, to a role anymore, the Admin has the task to create a new role's key, a new IK of that role and new re-encryption keys, and to send them to the appropriate recipient.

User Client. The User Client is the component of our applications that acts as the interface between the user and the RM running on the user's device. It takes user's commands and translates them into requests for the RM, whose response will be parsed and formatted in order to be displayed to the user. The User Client handles all the cryptographic actions of the user, in fact, it has the task to store and manage in a secure way the signing private key, the role's private key and the certificate of the user. Furthermore, it is tasked with the encryption and decryption of the data. How the client handles its user's cryptographic material is outside the scope of this thesis.

4.3 Cryptographic Scheme

In this section, we describe all the *cryptographic* steps that a resource passes through from the moment it is created and encrypted until it is decrypted. The symbols used in this section is shown in Table 4.2 and for a graphical representation of the cryptographic scheme, see Figure 4.3.

We assume that each user u possesses the following cryptographic material:

- a set containing secret-public key-pair for every role the user is assigned to [sk_{r_1}, pk_{r_1},..., sk_{r_n}, pk_{r_n}];
- a set containing all the public keys of the PDCs the user is assigned to pk_{PDC_1}, ..., pk_{PDC_n};
- signing key-pair (sk_u^{sig}, pk_u^{sig});

In our solution, we assign a pair of private-public keys to each PDC (sk_{PDC_n}, pk_{PDC_n}) during the creation steps of a PDC, see Section 4.4.3. These keys are then used — following the concept of hybrid encryption (see Section 3.2) — to encrypt all resources shared within the PDC. Afterwards, as depicted in Figure 4.3, the process for sharing a resource (assume it to be a plaintext m) through PDC_n begins with the encryption of m with a newly generated symmetric key $\mathbf{Gen^{Sym}} \rightarrow k^{sym}$, resulting in a ciphertext $\mathbf{Enc}^{\mathbf{Sym}}_{k^{sym}}(m) \rightarrow c$ (step 1 in Figure 4.3). Then, k^{sym} is encrypted with the public key of PDC_n (pk_{PDC_n}) and sk_u^{sig}, resulting in $\mathbf{Enc}^{\mathbf{Pub}}_{pk_{PDC_n}}(k^{sym}) \rightarrow \mathbf{K}$ (2.). From now on, only the administrators of the organizations in possession of sk_{PDC_n} are able to decrypt $\mathbf{K}$, limiting the total control over resources to the blockchain. $\mathbf{K}$ is now sent to the RM (3.), which has the task to check the permissions through the policies (4.), subsequently the RM invokes the appropriate smart-contract on the network in order to share $\mathbf{K}$ and c (5.). On the other side, a user associated with role r_n sends a request to the RM to obtain the resource (6.). The RM proceeds to checking the permissions (7.) and invoking the smart-contract on the network to fetch the asset with $\mathbf{K}$ and c. In order to share the resource with authorized users within each of these organizations, we rely on PRE (see Section 3.3). This approach allows a proxy entity to dynamically re-encrypt $\mathbf{K}$ in order to grant access to specific recipients as needed, without exposing k^{sym} to the proxy entity. In our solution, we consider RBAC as AC enforced by a CAC mechanism (see Section 3.4.1), therefore our recipients are the roles, in this case role r_n. This step includes the collaboration of more components: *admin*, RM and IP. The *admin* is in charge of creating an intermediary key $\mathbf{IK}_{r_n}$ for every role. Then, the *admin* creates $\mathbf{dk}_{sk_{PDC_n} \rightarrow \mathbf{IK}_{r_n}}$ that will be given to the RM and then $\mathbf{dk}_{\mathbf{IK}_{r_n} \rightarrow pk_{r_n}}$ that will be shared with the IP. $\mathbf{IK}_{r_n}$ consists of an asymmetric key-pair made out of a private and a

public key; the public key is used to create $\mathbf{dk}_{sk_{PDC_n} \to \mathbf{IK}_{r_n}}$, while the private key is used to create $\mathbf{dk}_{\mathbf{IK}_{r_n} \to pk_{r_n}}$. For semplicity, we keep the notation $\mathbf{IK}_{r_n}$. When RM receives $\mathbf{K}$, it performs the first PRE $\mathbf{PRE}_{dk_{sk_{PDC_n} \to \mathbf{IK}_{r_n}}}(\mathbf{K})$ obtaining $\mathbf{K}^{int}$ (9.). Then, RM sends $\mathbf{K}^{int}$ to IP (10.) which can continue performing the PRE $\mathbf{PRE}_{dk_{\mathbf{IK}_{r_n} \to pk_{r_n}}}(\mathbf{K}^{int})$, obtaining $\mathbf{K}^{end}$ (11.) and sending $\mathbf{K}^{end}$ back to the RM (12.). After the second re-encryption, RM provides the user with $\mathbf{K}^{end}$ and c (13.). At this point the user is able to decrypt $\mathbf{K}^{end}$ with sk_{r_n}, by performing $\mathbf{Dec}^{\mathbf{Pub}}_{sk_{r_n}}(\mathbf{K}^{end})$ and obtaining k^{sym} (14.). With $k^{sym}$the user is able to decrypt c obtaining the plain-text m (15.).

4.4 Basic operational flows

In this Section we describe the basic operational workflows of the system: creation and reading of an asset, setup of a new PDC and key revocation process.

4.4.1 Creation of an asset

The user u assigned to role *associate* wants to share a resource on a PDC, in this case PDC_1. In order to do that, u must first authenticate themself, so sends the request of authentication to the RM of their organization. The RM checks the request and, if valid, accepts the authentication of u. u at this point can create the asset and sends the request to the RM for the creation of the asset on PDC_1. The request can be abstractly represented as:

$$\texttt{<create, asset, PDC1, dataType>} \quad \text{where} \quad \text{asset} = \langle c, \mathbf{K} \rangle \tag{4.2}$$

Table 4.2: **Notations used to describe cryptographic operations**

Symbol	Description
u	User
$admin$	Admin
sk_u	Secret key of u
pk_u	Public key of u
sk_u^{sig}	Secret signing key of u
pk_u^{sig}	Public signing key of u
sk_{r_n}	Secret key of Role n
pk_{r_n}	Public key of Role n
sk_{PDC_n}	Secret key of PDC_n
pk_{PDC_n}	Public key of PDC_n
m	Plaintext
c	Ciphertext
k^{sym}	Symmetric key
$\mathbf{Gen}^{\mathbf{Sym}}$	Generation of the symmetric key k^{sym}
$\mathbf{Enc}^{\mathbf{Sym}}_{k^{sym}}$	Symmetric Encryption using k^{sym}
$\mathbf{Enc}^{\mathbf{Pub}}_{pk_{PDC_n}}$	Asymmetric Encryption using pk_{PDC_n}
$\mathbf{Dec}^{\mathbf{Pub}}_{sk_{r_n}}$	Asymmetric Decryption using sk_{r_n}
$\mathbf{K}$	Encrypted k^{sym}
$\mathbf{K}^{int}$	$\mathbf{K}$ after $\mathbf{PRE}_{dk_{sk_{PDC_n} \to \mathbf{IK}_{r_n}}}(\mathbf{K})$
$\mathbf{K}^{end}$	$\mathbf{K}^{int}$ after $\mathbf{PRE}_{dk_{\mathbf{IK}_{r_n} \to pk_{r_n}}}(\mathbf{K}^{int})$
$\mathbf{IK}_{r_n}$	Intermediary Key of Role n
$\mathbf{dk}_{sk_{PDC_n} \to \mathbf{IK}_{r_n}}$	Delegation Key that allows the Re-encryption from sk_{PDC_n} to $\mathbf{IK}_{r_n}$
$\mathbf{dk}_{\mathbf{IK}_{r_n} \to pk_{r_n}}$	Delegation Key that allows the Re-encryption from $\mathbf{IK}_{r_n}$ to sk_{r_n}
$\mathbf{PRE}_{dk_{sk_{PDC_n} \to \mathbf{IK}_{r_n}}}(\mathbf{K})$	Re-encryption of $\mathbf{K}$, which would only be decrypted with sk_{PDC_n} with delegation key $\mathbf{dk}_{sk_{PDC_n} \to \mathbf{IK}_{r_n}}$, thus making $\mathbf{K}$ decryptable by $\mathbf{IK}_{r_n}$
$\mathbf{PRE}_{dk_{\mathbf{IK}_{r_n} \to pk_{r_n}}}(\mathbf{K}^{int})$	Re-encryption of $\mathbf{K}^{int}$, which would only be decrypted with $\mathbf{IK}_{r_n}$ with delegation key $\mathbf{dk}_{\mathbf{IK}_{r_n} \to pk_{r_n}}$, thus making $\mathbf{K}^{int}$ decryptable by sk_{r_n}

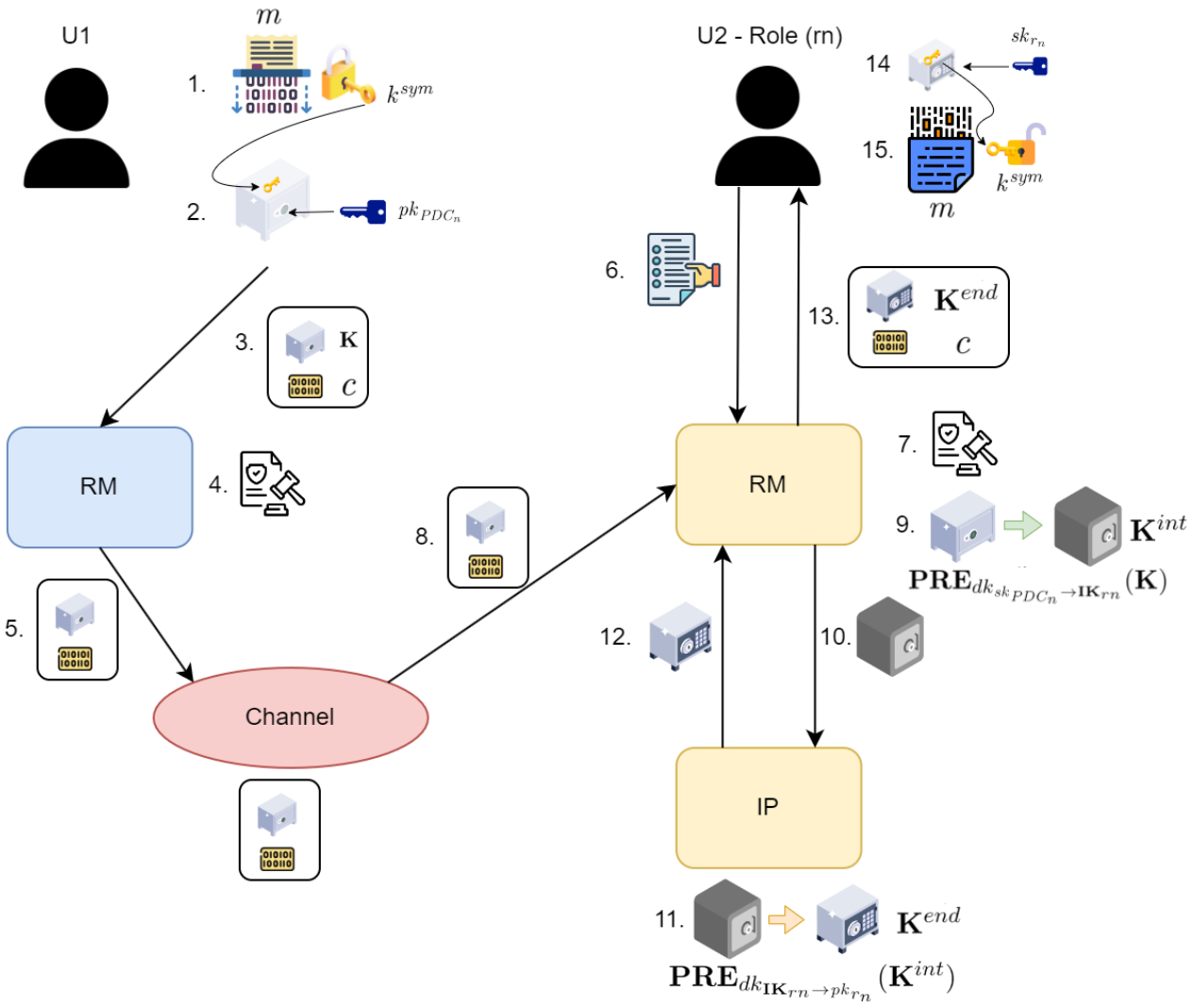

Figure 4.3: Cryptographic Scheme, sharing a resource over the network between two users belonging to different organizations

where c is the ciphertext and $\mathbf{Enc}^{\mathbf{Pub}}_{pk_{PDC_1}}(k^{sym}) \rightarrow \mathbf{K}$.

The RM receives the request, parses it and checks if in its ACL is present the policy <*associate*, CREATE, PDC_1, dataType>. If u is allowed to perform the operation, the RM will invoke the CREATE chaincode on the peer of the organization, which will share the asset on PDC_1. See Figure 4.4.

4.4.2 Reading of an asset

The user u assigned to role *associate* wants to read a resource on a PDC, in this case PDC_1. In order to do that, u must first authenticate themself, so sends the request of authentication to the RM of their organization. RM checks the request and, if valid, accepts the authentication of u. u can now sends a READ request to the RM. The RM parses it and checks if in its ACL is present the policy <*associate*, READ, PDC_1, dataType>. If u is allowed to perform the operation, the RM will invoke the READ chaincode on the peer of the organization. Now the RM fetches the request asset <`encrypted data`, $\mathbf{K}$, > from the peer and performs the first re-encryption $\mathbf{PRE}_{dk_{sk_{PDC_1}} \rightarrow \mathbf{IK}_{associate}}(\mathbf{K})$ obtaining $\mathbf{K}^{int}$. The RM sends $\mathbf{K}^{int}$ to IP. IP performs the second re-encryption $\mathbf{PRE}_{dk_{\mathbf{IK}_{associate}} \rightarrow pk_{associate}}(\mathbf{K}^{int})$ obtaining $\mathbf{K}^{end}$. Now the IP sends $\mathbf{K}^{end}$ back to the RM which will provide to sends it to u. u can decrypts the resource with $sk_{associate}$ by performing $\mathbf{Dec}^{\mathbf{Pub}}_{sk_{associate}}$. See Figure 4.5.

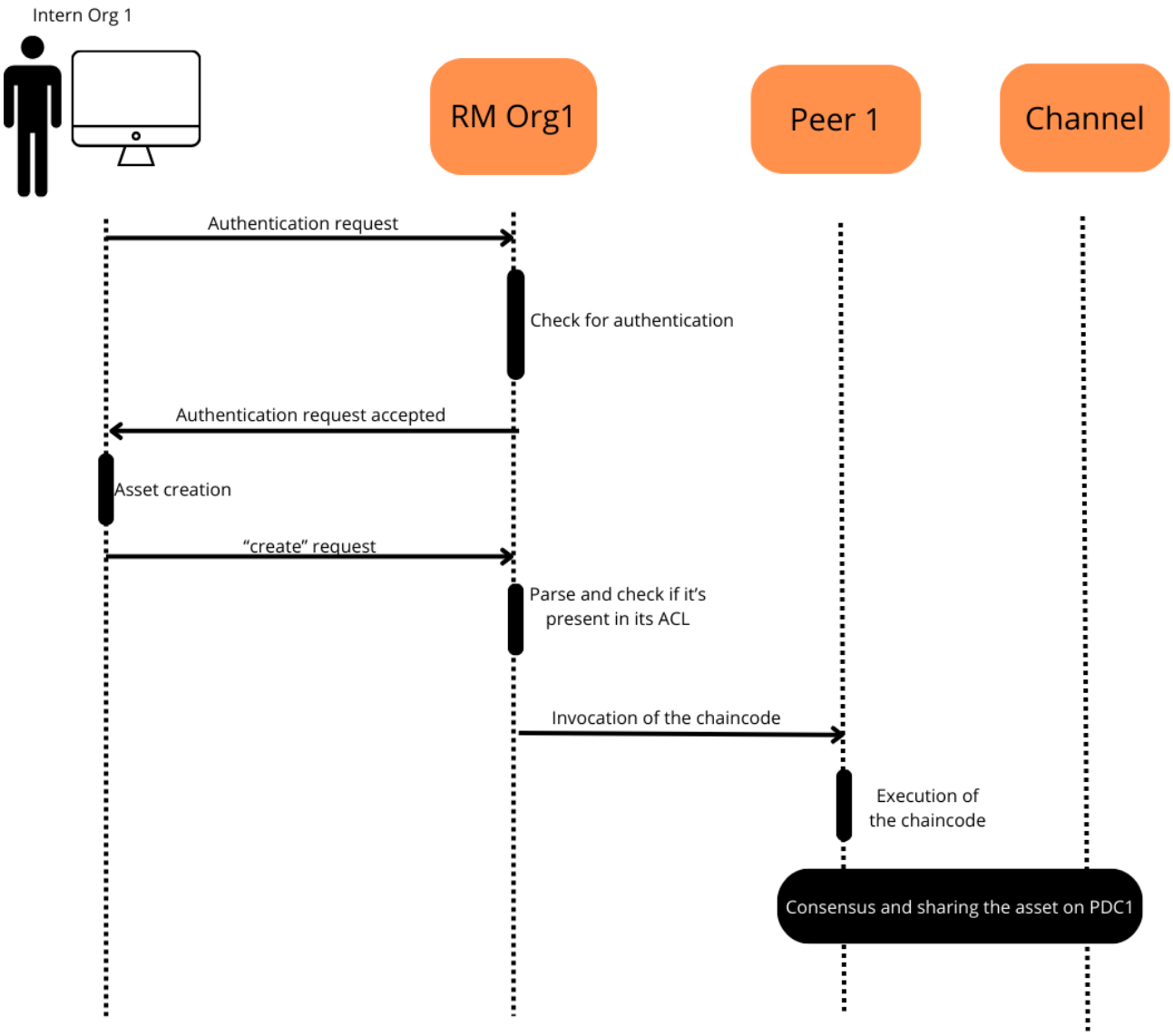

Figure 4.4: Creation of an asset

4.4.3 Setup of a new PDC

A PDC allows a subset of organizations to share securely data of the same type among themselves without the non-partecipating organizations being able to see the exchanged data. We point out that the types of data, the partecipating organizations and the rules governing data sharing are established through a contract agreed upon by the organizations. The data exchanged on a PDC can have more distinction specifying the dataType on them. The dataType permits a further fine-grained access control on data. It is important to notice the level of the security of the data in the system, in fact all the data shared in a PDC have a high level of security in terms of confidentiality, absence of collusion and accessibility. Nobody outside of the PDC can have access to the data. Instead, if the organizations want to add some dataType the solution can not provide the same level of security on them. So it is important to determine the level of security we want for that type of data and evaluate whether if it is advisable to create a new PDC or add new dataTypes.

In order to create a new PDC the partecipating organizations must first agree on the types of data to be shared on the PDC and their rules through a contract. Then, only one admin of one of the partipating organizations has to follow two main steps **Configuration on Hyperledger Fabric** and **Cryptographic Setup** to setup a new PDC in the system.

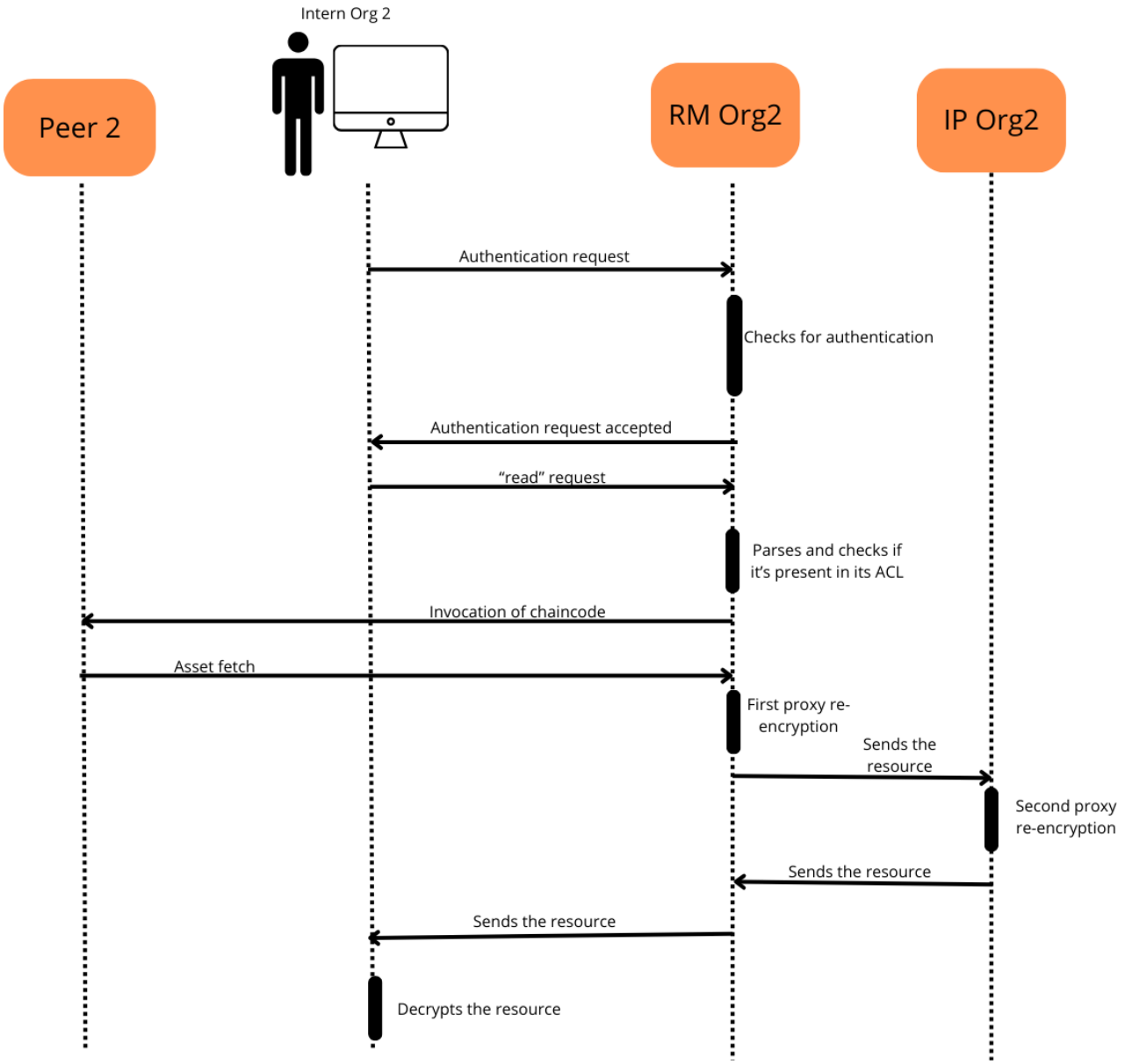

Figure 4.5: Reading of an asset

Configuration on Hyperledger Fabric The selected admin must first configurate the new PDC on the blockchain network. This main steps includes:

- Defining the PDC file that describe the PDC
- Updating the Chaincode on the Peer of its organization
- Deploying the Chaincode and instantiate the Chaincode on the network
- Updating the channels policies

At this point the PDC is instantiated on the network and all the peers of the partecipating organizations can interact with the PDC.

Cryptographic Setup As pictured in Figure 4.6 the selected admin creates a pair of key sk_{PDC_1} and pk_{PDC_1} for the PDC (1.). The admin encrypts sk_{PDC_1} with the public keys of the other participating organizations (2.) and shares to the network the set of encrypted private keys and pk_{PDC_1} (3.). The admins of the participating organizations fetch the appropriate encrypted key and pk_{PDC_1} from the network (4.) and decrypt the encrypted key with the organisation's private key (5.). At this point

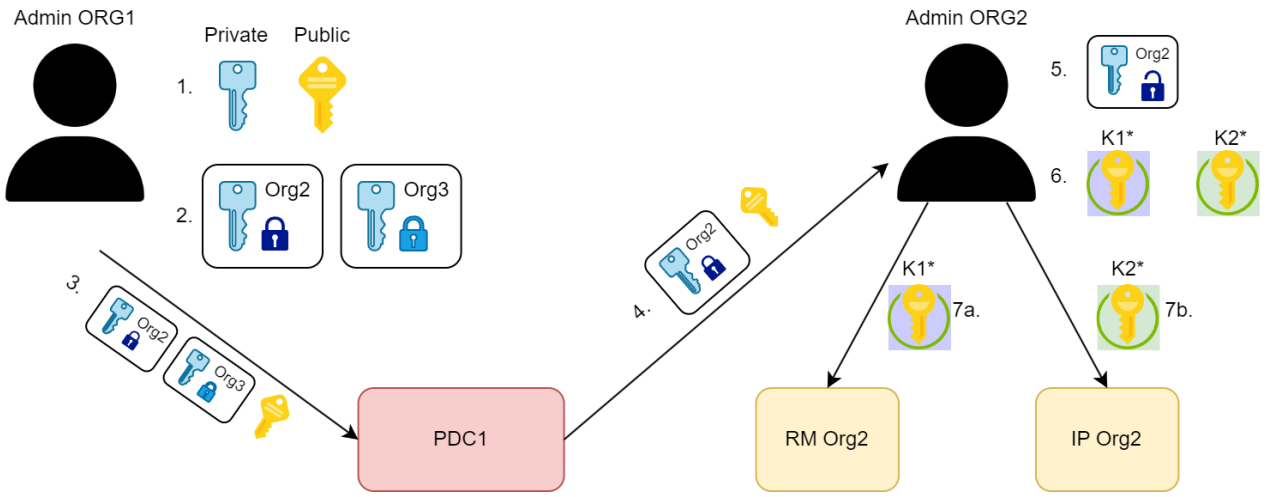

Figure 4.6: Creation of a PDC

each admin of the partecipating organizations creates the policies for the PDC deciding which roles can interact and which permissions they have, then updates the ACL of RM. In order to allow the RM and the IP to perform the PRE, the admin must now create a $\mathbf{IK}_{r_n}$ for each role that has access to the PDC, and the further delegation keys $\mathbf{dk}_{sk_{PDC_1} \to \mathbf{IK}_{r_n}}$ and $\mathbf{dk}_{\mathbf{IK}_{r_n} \to pk_{r_n}}$ (6.) to send respectively to RM and IP (7a. and 7b.).

4.4.4 Key Revocation

Key Revocation is a process used in cryptographic systems to invalidate a cryptographic key, making it unusable. In an organization it is possible that over time a user may lose their permissions for a certain role. For example, we can see in the Figure 4.7, U2 loses the permission for R1. This means that the user can no longer access to the resources associated to that role and decrypts them with the private role's key sk_{r_n}. Being that the user is in possession of the secret role's key sk_{r_n} and can save it on their own, deleting sk_{r_n} from the User Client of the user may not be enough, because the user can use the saved key for access to the unathorized resouces, possibly collouding with honest-but-curious entities such as RM and/or IP. In order to solve part of the problem the admin needs to creates the following cryptographic material (refering to the Figure 4.7):

- A new key-pair for the role $(sk^+_{R_1}, pk^+_{R_1})$
- A new IK for the role $(\mathbf{IK}^+_{R_1})$
- New delegations keys $(\mathbf{dk}_{sk_{PDC_1} \to \mathbf{IK}^+_{R_1}})$ and $(\mathbf{dk}_{\mathbf{IK}^+_{R_1} \to sk^+_{R_1}})$

At this point the admin shares $sk^+_{R_1}$ with all the users assigned to that role, excluding the revoked user. Then, the admin deletes from the lists of delegation keys the old delegation keys ($\mathbf{dk}_{sk_{PDC_1} \to \mathbf{IK}_{R_1}}$ and $\mathbf{dk}_{\mathbf{IK}_{r_n} \to sk_{R_1}}$) and adds the new delegation keys.

It remains the possibility that RM and IP, as honest-but-curious entities, can be corrupted and save the old delegation keys sharing the resources with the unauthorized user that is in possession of the old role's key. In order to do this, both RM and IP should be corrupted, if one of the two components is not corrupted the user cannot decrypt the resource with the old key.

We want to emphasize the importance of the IP and IK during the revocation of a user. In fact, if we have a single key-pair for every PDC, then all the resources shared on the PDC would be encrypted with the same public key and decrypted with the same secret key. By using the approach of PRE, the admin creates a delegation key that allows the RM to act as a proxy and transform the resource so it can be decrypted by the authorized role. In this case we would have a problem: in case of revocation, it is necessary to change not only the role's key and the delegation key, but also the private key of the PDC, that brings consequences to all the participating organizations. Instead, by introducing IP and IK we have to create only the cryptographic material listed before and we do not have the problem

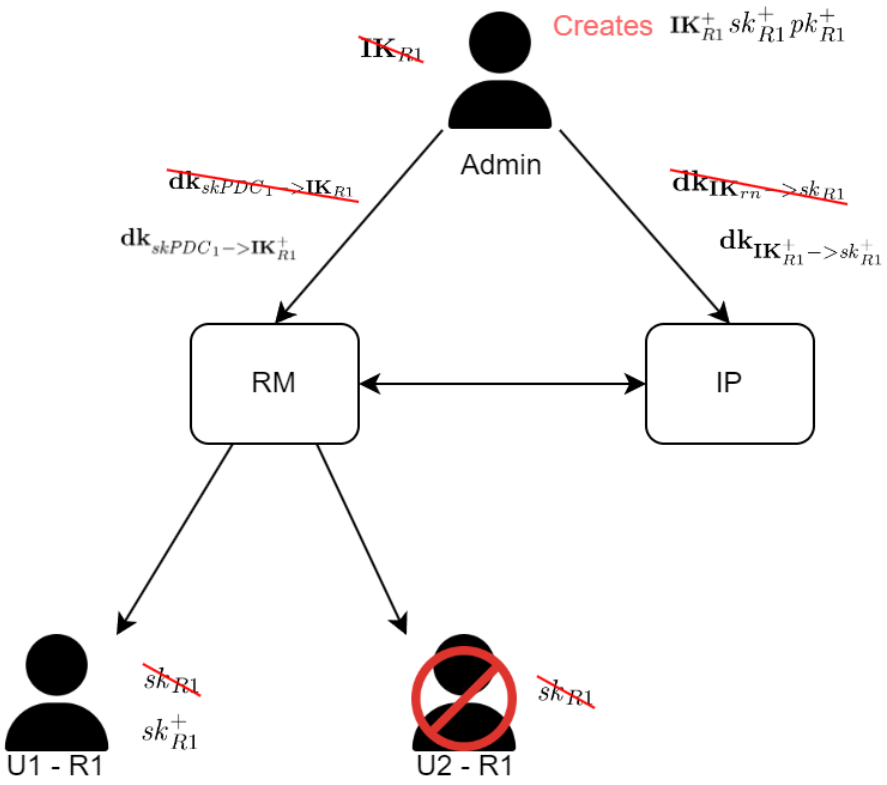

Figure 4.7: Revocation

of changing an enormous quantity of keys and overall the revocation brings consequences only for the interested organization and not for the whole network.

Re-encryption of resources The problem remains with all the resources of the PDC that the user was able to access and save the symmetric key to decrypt them, because these resources can still be decrypted by the user with those symmetric keys. In this case we can follow different procedures:

- **Instantaneous re-encryption** Re-encrypt all the resources shared in the PDC creating new symmetric keys, this procedure is very impactful;
- **No re-encryption** We rely on the AC mechanism to prevent unauthorized accesses and we leave the resources unchanged;
- **Lazy re-encryption** Re-encrypt the resources when the computational overhead and the loss of availability are less impactful.

In our architecture we employ the lazy re-encryption procedure.

5 Implementation

In this chapter, we present our proof of concept that we developed to demostrate the feasibility of our solution. The repository of the project and the instruction for the use can be found at [1]. In Section 5.1 we describe the configuration of the blockchain network used in the proof-of-concept, while in Section 5.2 we introduce the components of each organization and their functions.

5.1 Blockchain Network

A main component of our solution is the Blockchain Network. As our blockchain framework we decided to use HF. HF offers the possibility to create the channel, the PDCs, the organizations and the users that constitute the network and gives the tools to create all the certificate to authenticate the users. For the installation and configuration of HF we followed the guidelines present on the Hyperledger Fabric Docs[1]. As in the scenario, there are three organizations (Org1, Org2, Org3) that interact with the blockchain network, where each organization has its own peer that is responsible for mantaining the ledger and executing smart contracts. All the different elements of the blockchain run within Docker[2] containers that simulate a cloud environment, providing a consistent and isolated execution setup similar to that of a CSP. As for the current state of the peer's ledger, the world state, we decided to use CouchDB[3] as its state database, which stores assets as JSON objects that can be queried from the peer with ease. Following the scheme of the proposed scenario, we implemented the PDCs (PDCPatientInformation, PDCTreatmentInformation and PDCNeedsAnalysis) with the relative chaincodes. All the chaincodes are written in the Go programming language[4]. Every chaincode accepts one of the CRUD action and an asset consists of: *AssetID*, *AssetKey* and *AssetText*, where the *AssetID* is the unique identifier of the asset, the *AssetKey* stores the encrypted k^{sym} and the *AssetText* stores the encrypted data.

5.2 CryptoGate

In our proof-of-concept the components of each organization are defined in a project — called CryptoGate. All the code of CryptoGate is written in JavaScript programming language[5] with Node.js[6] as it is a flexible language and compatible with both Hyperledger Fabric (for the implementation of the Fabric Gateway[7] with the Hyperledger Fabric SDK for Node.js[8]) and the library recrypt-rs of Ironcorelabs[9]. CryptoGate is subdivided into the main components: RM, IP, userClient and admin. Each component has its own server implemented in Express.js[10] and interacts through API requests with the other components replicating a cloud environment. In the proof-of-concept we decide not to implement the MM since it is an important but not a core component of the solution.

UserClient UserClient is responsible for forwarding the user request to the RM and for fetching the response. The user can send the request through a web interface compiling the module as in the Figure 5.1. This component can access to the user's keys and carry out the encryption and decryption of the k^{sym} and the data. The symmetric encryption is performed using the AES-256-GCM algorithm using functions of the library crypto[11] while the asymmetric encryption is performed using the function *encryption* of the library recrypt-rs and the decryption with the function *decryption* of the library recrypt-rs.

Reference Monitor The RM is the component that receives the request from UserClient, checks the policies and forwards the data to the Blockchain Network. The policies are stored in a JSON file as `[role, action, PDC, dataType]`. The connection with the Blockchain Network is through the implementation of the Farbic Gateway, that permits the interaction with the blockchain network. Then, the RM is responsible for performing the first re-encryption on the data stored on the Blockchain Network, this is done using the function *transfrom* of the the library recrypt-rs.

1 `https://hyperledger-fabric.readthedocs.io/en/latest/`
2 `https://www.docker.com/`
3 `https://couchdb.apache.org/`
4 `https://go.dev/`
5 `https://www.javascript.com/`
6 `https://nodejs.org/en`
7 `https://hyperledger-fabric.readthedocs.io/en/latest/gateway.html`
8 `https://hyperledger.github.io/fabric-sdk-node/`
9 `https://github.com/IronCoreLabs/recrypt-node-binding`
10 `https://expressjs.com/`
11 `https://nodejs.org/api/crypto.html`

Figure 5.1: User interface of CryptoGate

Intermediary Proxy The IP fetchs the data from RM after the first re-encryption through API and performs the second re-encryption using the function *transfrom* of the the library recrypt-rs with the appropriate delegationKey.

Admin The admin is the component in possess of all the keys and is responsible for generating new key pair by executing the function *generateKeyPair* of the library recrypt-rs, for generating the delegation keys to send to the RM and IP by executing the function *generateTransformKey* of the library recrypt-rs and for creating and manage the intern AC policies of the organization.

6 Conclusion

In this thesis, we propose a solution for guaranteeing the confidentiality of sensitive data shared over a Cloud-managed blockchain network managed by a partially trusted CSP in cross-organizational scenarios. This project builds upon a previous solution developed by Enrico Marconi [11]. During the analysis we identified some limitations and shortcomings, and in our work we have addressed and overcome these issues. The result of this process comes out as a permissioned blockchain-based system that enables different organizations to securely and selectively share sensitive data among each other in a cloud environment. The system benefits from the advantages of both blockchain and cloud technologies. Blockchain provides properties such as authentication, integrity, and decentralization, while the cloud addresses challenges related to setup, maintenance, and usability of the blockchain. Additionally, the cloud offers other benefits like scalability, accessibility, and cost efficiency. A cloud-managed permissioned blockchain, however, cannot ensure an high level of confidentiality of the data,

since the cloud must be considered as a partially-trusted entity and, as the CSP handles the nodes composing the blockchain, it has ideally access to the data contained within the transactions. All the data managed by the blockchain must be made inaccessible, therefore, in our solution we designed a cryptographic scheme that encrypts all the data shared on the blockchain and enables the organizations to enforce fine-grained RBAC policies on the shared data so that data can be accessed only by the authorized users. The cryptographic construction we developed for this matter combines symmetric and asymmetric encryption techniques combining with a PRE approach. As a whole, this construction allows organizations to privately share sensititive data among other with an high level of confidentiality, without the need to disclose the inner structure of the other organizations. Through the use of PRE approach the system is able to efficiently handle the re-keying or revocation processes and to avoid collusions between authorized and unauthorized users and untrusted entities. Finally, to demonstrate the feasibility of the solution, we have implemented a proof-of-concept [1] with the blockchain network developed using HF and the other components of the solution written in Javascript, providing a web interface to interact with the network.

Future Work The implementation's code could be further optimized and completed. The proof-of-concept is written in JavaScript, we plan to implement the solution also in programming language such as Java, since Java is a more structured language and more appropriate for high-security environments. Finally, our solution could also be adapted to make each organization able to enforce any AC method, e.g. ABAC.

Bibliography

[1] Source code repository for proof-of-concept implementation. `https://gitlab.fbk.eu/st/people/students/2024-luca-claus/documents/-/tree/main/Project?ref_type=heads`.

[2] Kwame Opuni-Boachie Obour Agyekum, Qi Xia, Emmanuel Boateng Sifah, Christian Nii Aflah Cobblah, Hu Xia, and Jianbin Gao. A proxy re-encryption approach to secure data sharing in the internet of things based on blockchain. *IEEE Systems Journal*, 16(1):1685–1696, 2022.

[3] Stefano Berlato, Roberto Carbone, Adam J. Lee, and Silvio Ranise. Formal modelling and automated trade-off analysis of enforcement architectures for cryptographic access control in the cloud. *ACM Trans. Priv. Secur.*, 25(1), nov 2021.

[4] William C. Garrison, Adam Shull, Steven Myers, and Adam J. Lee. On the practicality of cryptographically enforcing dynamic access control policies in the cloud. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 819–838, 2016.

[5] Shadan Ghaffaripour and Ali Miri. Cryptographically enforced access control in blockchain-based platforms. In *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–7, 2019.

[6] Matt Gorbett, Hossein Shirazi, and Indrakshi Ray. Local intrinsic dimensionality of iot networks for unsupervised intrusion detection. In *Data and Applications Security and Privacy XXXVI: 36th Annual IFIP WG 11.3 Conference, DBSec 2022, Newark, NJ, USA, July 18–20, 2022, Proceedings*, page 143–161, Berlin, Heidelberg, 2022. Springer-Verlag.

[7] Andreas Heider-Aviet, Danny Roswin Ollik, Stefano Berlato, Silvio Ranise, Roberto Carbone, Van Thanh Le, Nabil El Ioini, Claus Pahl, and Hamid R. Barzegar. Blockchain based ran data sharing. In *2021 IEEE International Conference on Smart Data Services (SMDS)*, pages 152–161, 2021.

[8] IronCore Labs. Transform encryption. 2022.

[9] Hsiao-Ying Lin, John Kubiatowicz, and Wen-Guey Tzeng. A secure fine-grained access control mechanism for networked storage systems. In *2012 IEEE Sixth International Conference on Software Security and Reliability*, pages 225–234, 2012.

[10] Ahsan Manzoor, Madhsanka Liyanage, An Braeke, Salil S. Kanhere, and Mika Ylianttila. Blockchain based proxy re-encryption scheme for secure iot data sharing. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 99–103, 2019.

[11] Enrico Marconi. Combining blockchain-as-a-service and cryptographic access control for secure data sharing across multiple organizations. *Bachelor's Thesis of University of Trento*, 2021/2022.

[12] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[13] Uchi Ugobame Uchibeke, Kevin A. Schneider, Sara Hosseinzadeh Kassani, and Ralph Deters. Blockchain access control ecosystem for big data security. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1373–1378, 2018.

[14] Bob Wall and Patrick Walsh. Cryptographically enforced orthogonal access control at scale. In *Proceedings of the 6th International Workshop on Security in Cloud Computing*, SCC '18, page 57–65, New York, NY, USA, 2018. Association for Computing Machinery.

[15] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *2010 Proceedings IEEE INFOCOM*, pages 1–9, 2010.

[16] Guy Zyskind, Oz Nathan, and Alex 'Sandy' Pentland. Decentralizing privacy: Using blockchain to protect personal data. In *2015 IEEE Security and Privacy Workshops*, pages 180–184, 2015.