Universidade de Aveiro
2013
Departamento de
Electrónica, Telecomunicações e Informática

Tiago Miguel
Ferreira Guimarães
Pedrosa

Electronic Health Records for Mobile Citizens: A Secure and Collaborative Architecture

Uma arquitectura segura e colaborativa para Registos de Saúde Electrónicos com suporte a mobilidade

Universidade de Aveiro
2013
Departamento de
Electrónica, Telecomunicações e Informática

Tiago Miguel Ferreira Guimarães Pedrosa

Electronic Health Records for Mobile Citizens: A Secure and Collaborative Architecture

Uma arquitectura segura e colaborativa para Registos de Saúde Electrónicos com suporte a mobilidade

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Informática, do Programa Doutoral em Informática conjunto das Universidades do Minho, Aveiro e Porto – MAPi, realizada sob a orientação científica do Doutor José Luís Oliveira, Professor Associado da Universidade de Aveiro e do Doutor Rui Pedro Lopes, Professor Coordenador do Instituto Politécnico de Bragança.

o júri / the jury

presidente / president

Doutor Fernando Joaquim Tavares da Rocha
Professor Catedrático da Universidade de Aveiro

Doutor Gabriel de Sousa Torcato David
Professor Associado da Faculdade de Engenharia da Universidade do Porto

Doutor Alexandre Júlio Teixeira Santos
Professor Associado da Escola de Engenharia da Universidade do Minho

Doutor Paulo José Osório Rupino da Cunha
Professor Auxiliar da Faculdade de Ciências da Universidade de Coimbra

Doutor António Manuel de Jesus Pereira
Professor Coordenador da Escola Superior de Tecnologia e Gestão de Leiria do Instituto Politécnico de Leiria

Doutor José Luis Guimarães Oliveira
Professor Associado da Universidade de Aveiro (orientador)

Doutor Rui Pedro Sanches de Castro Lopes
Professor Coordenador da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Bragança (co-orientador)

agradecimentos / acknowledgements

Quero aproveitar esta oportunidade para agradecer aos meus orientadores o apoio e orientação permanente.

Uma palavra de agradecimento aos membros do fabuloso grupo UA.PT Bioinformatics pelo acolhimento, discussões, contributos científicos e atividades lúdicas que me ajudaram a manter um pouco de sanidade.

A todos os colegas do MAP-i com quem trabalhei, discuti e aprendi, o meu obrigado. Aos docentes das Universidades envolvidas com quem tive o prazer de trabalhar, um bem hajam por terem tornado esta experiência ainda mais enriquecedora.

Gostaria de agradecer ao Instituto Politécnico de Bragança pelas condições que me proporcionou, bem como a todos os colegas que de uma forma ou outra me apoiaram durante esta caminhada.

À Fundação para a Ciência e a Tecnologia (Programa PROTEC, bolsa SFRH/BD/49765/2009) pelos apoios de vária ordem que me foram atribuídos.

Aos amigos e familiares agradeço o estímulo e a compreensão. Peço desculpa pelo pouco tempo que vos pude dispensar.

Finalmente, uma palavra especial para os meus pais, irmã, namorada, avós, madrinha e dois padrinhos: Muito obrigado por tudo!

Palavras-chave Registos de Saúde Electrónicos, Registos de Saúde Pessoais, Mobilidade, Colaboração, Arquitectura Orientada ao Serviço, Segurança

Resumo

Durante as últimas décadas, os registos de saúde electrónicos (EHR) têm evoluído para se adaptar a novos requisitos. O cidadão tem-se envolvido cada vez mais na prestação dos cuidados médicos, sendo mais pró activo e desejando potenciar a utilização do seu registo. A mobilidade do cidadão trouxe mais desafios, a existência de dados dispersos, heterogeneidade de sistemas e formatos e grande dificuldade de partilha e comunicação entre os prestadores de serviços.

Para responder a estes requisitos, diversas soluções apareceram, maioritariamente baseadas em acordos entre instituições, regiões e países. Estas abordagens são usualmente assentes em cenários federativos muito complexos e fora do controlo do paciente. Abordagens mais recentes, como os registos pessoais de saúde (PHR), permitem o controlo do paciente, mas levantam dúvidas da integridade clinica da informação aos profissionais clínicos. Neste cenário os dados saem de redes e sistemas controlados, aumentando o risco de segurança da informação. Assim sendo, são necessárias novas soluções que permitam uma colaboração confiável entre os diversos actores e sistemas.

Esta tese apresenta uma solução que permite a colaboração aberta e segura entre todos os actores envolvidos nos cuidados de saúde. Baseia-se numa arquitectura orientada ao serviço, que lida com a informação clínica usando o conceito de envelope fechado. Foi modelada recorrendo aos princípios de funcionalidade e privilégios mínimos, com o propósito de fornecer protecção dos dados durante a transmissão, processamento e armazenamento. O controlo de acesso é estabelecido por políticas definidas pelo paciente. Cartões de identificação electrónicos, ou certificados similares são utilizados para a autenticação, permitindo uma inscrição automática. Todos os componentes requerem autenticação mútua e fazem uso de algoritmos de cifragem para garantir a privacidade dos dados. Apresenta-se também um modelo de ameaça para a arquitectura, por forma a analisar se as ameaças possíveis foram mitigadas ou se são necessários mais refinamentos.

A solução proposta resolve o problema da mobilidade do paciente e a dispersão de dados, capacitando o cidadão a gerir e a colaborar na criação e manutenção da sua informação de saúde. A arquitectura permite uma colaboração aberta e segura, possibilitando que o paciente tenha registos mais ricos, actualizados e permitindo o surgimento de novas formas de criar e usar informação clínica ou complementar.

Keywords Electronic Health Records, Personal Health Records, Mobility, Collaboration, Service Oriented Architecture, Security

Abstract

Since their early adoption Electronic Health Records (EHR) have been evolving to cope with increasing requirements from institutions, professionals and, more recently, from patients. Citizens became more involved demanding successively more control over their records and an active role on their content. Mobility brought also new requirements, data become scattered over heterogeneous systems and formats, with increasing difficulties on data sharing between distinct providers.

To cope with these challenges several solutions appeared, mostly based on service level agreements between entities, regions and countries. They usually required defining complex federated scenarios and left the patient outside the process. More recent approaches, such as personal health records (PHR), enable patient control although raises clinical integrity doubts to other actors, such as physicians. Also, information security risk increase as data travels outside controlled networks and systems. To overcome this, new solutions are needed to facilitate trustable collaboration between the diverse actors and systems.

In this thesis we present a solution that enables a secure and open collaboration between all healthcare actors. It is based on a service-oriented architecture that deals with the clinical data using a closed envelope concept. The architecture was modeled with minimal functionality and privileges bearing in mind strong protection of data during transmission, processing and storing. The access control is made through patient policies and authentication uses electronic identification cards or similar certificates, enabling auto-enrollment. All the components require mutual authentication and uses cyphering mechanisms to assure privacy. We also present a threat model to verify, through our solution, if possible threats were mitigated or if further refinement is needed.

The proposed solution solves the problem of patient mobility and data dispersion, and empowers citizens to manage and collaborate in their personal healthcare information. It also permits open and secure collaboration, enabling the patient to have richer and up to date records that can foster new ways to generate and use clinical or complementary information.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ADL	Archetype Definition Language.
AES	Advanced Encryption Standard.
AOM	Archetype Object Model.
AQL	Archetype Query Language.
ASCII	American Standard Code for Information Interchange.
CA	Certificate Authority.
CDA	Clinical Document Architecture.
CDO	Care Delivery Organization.
CDR	Clinical Data Repository.
CDSS	Clinical Decision Support System.
CEN	European Committee for Standardization.
CMV	Controlled Medical Vocabulary.
CPOE	Computerized Provider Order Entry.
DAC	Discretionary Access Control.
DES	Data Encryption Standard.
DICOM	Digital Imaging and Communication in Medicine.
ebXML	Electronic Business using Extensible Markup Language.
EDIFACT	Electronic Data Interchange for Administration, Commerce and Transport.
EHR	Electronic Health Record.
eID	electronic Identification.
eMAR	electronic Medication Administration Record.
EMR	Electronic Medical Record.
epSOS	European Patient Smart Open Services.
hEHR	hybrid Electronic Health Record.
HIPAA	Health Insurance Portability and Accountability Act.
HL7	Health Level Seven.
HTML	HyperText Markup Language.
HTTP	Hypertext Transfer Protocol.
HTTPS	HyperText Transfer Protocol over Secure Socket Layer.
ICD	International Classification of Diseases.
ICEHR	Integrated Care EHR.

IETF	Internet Engineering Task Force.
IHE	Integrating the Healthcare Enterprise.
ISO	International Organization for Standardization.
LOINC	Logical Observation Identifiers Names and Codes.
MAC	Message Authentication Codes.
MAC	Mandatory Access Control.
MeSH	Medical Subject Headings.
MIME	Multipurpose Internet Mail Extensions.
MS-PDC	Multi-Service Patient Data Card.
PDF	Portable Document Format.
PHR	Personal Health Record.
PKI	Public-Key Infrastructure.
PRA	Patient Record Architecture.
RBAC	Role Based Access Control.
RID	Retrieve Information for Display.
RSBAC	Rule-Set Based Access Control.
SNOMED	Systemized Nomenclature of Medicine.
SOA	Service Oriented Architecture.
SR	Structured Reporting.
UID	Unique Identifier.
UMLS	Unified Medical Language System.
VHCS	Virtual Health Card Service.
VU-EPR	Virtual Unique Electronic Patient Card.
WADO	Web Access to DICOM Persistent Objects.
WSDL	Web Service Definition Language.
XDS	Cross-Enterprise Document Sharing.
XHTML	Extensible Hypertext Markup Language.
XML	Extensible Markup Language.

Chapter 1

Introduction

In the past decades, the increasing use of informatics technologies in the medical area lead to a new sub-area – the medical informatics – that is still in continuous evolution and trying to keep up with the huge demand of the medical area. Digital medical records have also been a hot research topic for many years, mainly due to paper-based records having reached their limits and due to new strategies to improve healthcare services. Additionally, sharing of information between professionals and patients also poses a problem. The routing overhead of paper based-records and their disadvantages demand the need of a digital form record. Digital records could enable the creation of new services that can make the best use of available information, such as supporting medical activities, decision support systems, automatic warnings for anomalous reactions of medicines and procedures for a specific patient, more involvement of the patient and care agents. Those are some of the reasons that sustain that digital medical records are a starting point to enable better care, since they can maintain and easily provide valuable data, enabling professionals to give proper treatments to citizens.

Fostering the goal for a usable and effective digital medical record, researchers and standardization bodies created three main types of records: i) Electronic Medical Record (EMR); ii) Electronic Health Record (EHR); iii)Personal Health Record (PHR). The EMR is oriented for use within a healthcare provider, the EHR is centered on the patient and tries to integrate information from the different providers, finally the PHR is patient centered and managed by the patient himself.

Different approaches were developed to solve specific requirements and to be used in well-defined scenarios. Unfortunately, providers choose different solutions and records tend to be incomplete and scattered among different providers during the life of the patient. To address this problem, interoperability between them is required. Despite of the functional level, at a semantic level the challenge is bigger because systems use different formats and codings. This aspect is hindering the creation, maintenance and use of a true integrated care EHR. This depends on the ability to integrate information from all healthcare providers.

As the number of providers and users increase, the complexity and challenges to maintain a secure and efficient infrastructure for digital medical records also increases. Therefore, the emergence of secure approaches which cope with active collaboration of all actors along with the patient is crucial. The openness for this collaboration has to be feasible as well with international providers, due to the increasing citizen mobility across countries. This was, as such, the initial context that has set this research in motion.

Digital medical records, in their EMR, EHR and PHR form, were specified some years

ago, based on requirements and assumptions which are continuously evolving. Citizens' way of living has suffered several and drastic modifications in recent years. These changes have also influenced the healthcare field, bringing new challenges to digital medical records. Therefore, this work starts with an analysis of how these records cope with new challenges.

The creation of a real longitudinal lifelong healthcare record demands the open collaboration between all healthcare providers and the patient himself. However, several issues must be solved first in order to to cope with this philosophy.

Using a high level systematic approach, some procedures must be followed bearing in mind a successful achievement:

- Definition of requirements for a collaborative record;
- Analysis of record types and formats to evaluate their adequacy in answering the identified requirements;
- Study and proposal of an infrastructure comprising core services for collaborative records, prepared to nurture future records utilization and new services development.

These procedures bring several challenges. A solution that needs to deal with collaboration of heterogeneous international healthcare providers, relies on the trust of those providers. Trust on the providers has to be enforced by the solution, without restricting the freedom of choice regarding the health provider by the patient. For expediting the bureaucracy involved in sharing access to patient medical records, the latter should be the responsible for managing his access control policy. Hence, the infrastructure should provide services for simplified management of access policy by the patient.

Medical information is one of the most sensitive data that systems can manipulate, i.e. security must be taken very seriously. The information resides in repositories which should be secure and protected. The interoperability requirements and the freedom of choosing any repository provider imply the use of standardized storage solutions. Data security cannot rely only on the storage service, so the infrastructure should provide services that could cope with requirements regardless of the chosen repository.

Using query functionalities surely improves the use of storage services, although they can also cause information leakage. This is due to the fact that being able to query and checking the number of matches at the same time can reveal information on the storage records. Therefore, the infrastructure should provide more secure query mechanisms that minimize the risk of direct or indirect information leaking.

This thesis tries to establish a solution that enables the creation and usage of a secure EHR, addressing issues such as citizen's mobility, actors collaboration and flexibility to evolve according to future requirements.

1.1 Motivation and Challenges

Electronic Health Records (EHR) can be defined as the whole data acquired during a patient's passage through the health care system that is maintained in a digital media. These records soon draw the attention of practitioners, policy makers and patients, since they can provide better integration between clinical services and enable health information sharing. However, EHRs have mainly been deployed in roughly enclosed health care scenarios. Strict regulatory, ethic and legal issues have hindered the wide adoption of EHRs, delaying the

establishment of a competitive market, where different providers could take full advantage of information exchange and regular practitioners' collaboration. Citizens are also demanding more control and access to their own personal medical data, as they become more informed.

PHR emanated to cope with these new user requirements to allow users to keep record of their own medical data. Several solutions have been developed, such as Google Health [13], Microsoft HealthVault [13] and Dossia [13]. These web-based PHRs are normally based on a central repository and set of core features. Some implementations are capable of extending features using external third-party services.

One of the big differences between EHR and PHR is related to the responsibility of maintaining the information and specifying the access control policy. In the former model healthcare institutions and their professionals are responsible for those operations. In the latter, the patient owns the rules. Despite this typical operational model, some PHR can also be automatically populated. The information can be fed through hospital information systems where the data is originally generated. However, to enable this scenario, it will be necessary for EHR systems to generate the adequate reports of clinical events and to interact directly with those external PHR systems. This will allow the patient to choose his PHR provider, to keep track of all personal health events without having to replicate the information manually and with the ability of controlling the access to the information. Digital medical records face challenges which are directly related to the changes both in the way of living and healthcare providing. New healthcare related services have emanated and will thus continue to appear. Some, already comprise a network of service providers and make use of already deployed systems for manipulating patient records data. Others lack these services and usually the access to patient medical record is not allowed. A solution that enables them to collaborate at the patient record level, would enable the patient to have an updated record which could be complemented with information related to health and not restricted to medical data.

Patients' mobility is also a challenging issue for digital medical records. Some level of sharing is already in place, but usually the health providers belong to the same organization or network of providers. Therefore, they can use the shared records in a simpler way than if the providers where from different healthcare networks, private or publics and even from different countries. This is the new reality, different providers provide simultaneous care to patients. Hence, all actors need to use patient records. The access to the record should be seamless to providers, enabling providers' easy access, which should not be a limiting factor when patient choose his/her health provider.

Considering these changes, how can already deployed systems cope with them? Briefly, EMR was created to enable a digital medical record inside a care delivery organization. It was not designed for sharing between different organizations or healthcare related professionals providers [1]. It was conceived to be used within organizations as a support for systems which facilitate healthcare delivering. On the other hand, EHR can support the sharing of patient information between different care providers. These records are patient centred, they can receive information directly from the clinical data repository. However, sharing is based on agreements between care providers which may thus limit the patient's free choice for a provider. If the patient wants an updated record and to grant access to his record, he can only choose providers that have agreements with the record holder organization.

A simplified approach is therefore PHR. This record approach gives patient full administration and data manipulation rights. Usually, data is created and maintained by the patient but it is hard to maintain an updated record. Another problem is the lack of standardization between records, since different solutions use different formats.

An updated record is possible to achieve if all the providers and patients can create and manipulate the data in a secure way, using information technologies standards to not exclude providers based on technological options. Besides, agreements with providers need to exist to export information from providers to the patient centralized record. This open collaboration brings challenges to the health record. It has to cope with data security, medical data integrity and semantic issues. Most of the already deployed solutions do not enable a real collaboration between actors. A certain type of collaboration can occur in well-defined groups, but a wider collaboration is rather limited. Therefore, an architecture to support this open collaboration is needed.

Data manipulated in these systems is very critical and personal and this is one of the main reasons why medical informatics is a good area to research about and to apply information security. The security requirements are pushed forward by the need of confidentiality, integrity and availability of the information combined with accountability of access and traceability of the information created.

Both the exchange and storage of health information are a major security challenge because its disruption may compromise personal privacy seriously. Albeit the legal constraints associated with this type of data are largely regulated [14, 15], they impose strict rules on the development of technical solutions. Moreover, having an enterprise with access to all health information of a citizen is unlikely to occur because the risk of information disclosure is likely to be higher and more disastrous than in a scattered scenario. In fact, the *Big Brother* scenario also emanates whenever the guardian of the information is an enterprise or the government. This vision does slow down the adoption of different solutions.

The open collaboration of all the actors, even from different countries, raises critical challenges. The desirable scenario is that every provider may collaborate to the patient record. These challenges are related to user authentication, authorization, accounting, secured communication between the different components of the architecture, secure storing and querying.

1.2 Problem statement

The fundamental problem statement in this issue is to propose an open and secure architecture that supports integrating health related information from several procedures and activities. This architecture is the foundation for collaboration from different healthcare and alternative medicine providers, well-being activities and others, under patient empowerment. This further enhances patient mobility, an obvious requirement in the modern way of life.

1.3 Contributions

The main contribution of this work is related to the way of coping with the challenges of creating digital medical records, namely facing issues such as mobility, openness and collaboration. Firstly, this research aims at understanding why the use of EHR and PHR are not widespread; secondly, it proposes a solution for current and future use; and finally it wishes to improve the security of such solution. To achieve it, preliminary objectives where outlined:

- To identify the types of digital medical records and new challenges they are facing;
- To propose a solution for coping with different challenges, based on standards and openness;

- To identify security requirements for the solution and propose an architecture supporting them;

- To discuss and analyse the proposed solution, including a threat model for a security analysis.

The solution should enable open and secure collaboration, based on open standards, among all actors involved in healthcare providing, where the access policy is defined by the citizen, without depending of previous agreements between the healthcare providers.

The work in this thesis has partly or fully contributed to the following publications:

- Tiago Pedrosa, Carlos Costa, José Luís Oliveira, Rui Pedro Lopes, "Virtual Health Card System", 1 Inforum 2009, University of Lisbon, Lisbon, Portugal, 10 and 11 of September of 2009

- João Santos, Tiago Pedrosa, Carlos Costa, José Luís Oliveira, "Modelling a Portable Personal Health Record", HealthInf 2010, Valencia, Spain, 20-23 January of 2010

- Tiago Pedrosa, Rui Pedro Lopes, João Santos, Carlos Costa, José Luís Oliveira, "Towards an EHR Architecture for Mobile Citizens", HealthInf 2010, Valencia, Spain, 20-23 January of 2010

- João Santos, Tiago Pedrosa, Carlos Costa, José Luís Oliveira, "Gathering and Managing Complementary y Diagnostic Tests", 5 Conferência Ibérica de Sistemas e Tecnologias de Informação, Santiago Compostela, Spain, June 2010

- João Santos, Tiago Pedrosa, Carlos Costa, José Luís Oliveira, "Portable Personal Health Records: Breaking The Glass Of Emergency Access To Data", 6 Conferência Ibérica de Sistemas e Tecnologias de Informação, Chaves, Portugal, June 2011

- João Santos, Tiago Pedrosa, Carlos Costa, José Luís Oliveira, "On The Use of OpenEHR in a Portable PHR", HealthInf 2011, Rome, Italy, 26-29 January 2011

- Tiago Pedrosa, Rui Pedro Lopes, João Santos, Carlos Costa, José Luís Oliveira, "Hybrid Electronic Health Records", HealthInf 2011, Rome, Italy, 26-29 January 2011

- José Luís Oliveira, Tiago Pedrosa, Carlos Costa, Linda Velte, "An OpenEHR Repository Based on a Native XML Database", HealthInf 2012, Vilamoura, Portugal, 1-4 February 2012

- Tiago Pedrosa, Rui Pedro Lopes, João Santos, Carlos Costa, José Luís Oliveira, "A Secure Personal Health Record Repository", HealthInf 2012, Vilamoura, Portugal, 1-4 February 2012

- Cândido Santos, Tiago Pedrosa, Carlos Costa, José Luís Oliveira, "Concepts for a Personal Health Record", MIE 2012, Pisa, Italy, 26-29 August

1.4 Thesis Outline

Chapter 2 provides a background analysis concerning the type of records, EHR systems, standards, and regulatory framework. Some solutions for integrating medical records from different providers are studied. Security requirements and methods for regulating access to the records are also studied and discussed. During this chapter, a critical analysis is made contemplating how the several approaches cope with the new challenges.

In chapter 3 we define the requirements for such a type of solution and propose an architecture which can cope with them.

Chapter 4 provides a validation of the proposed model, verifying the addressing of the requirements, namely a security analysis.

Chapter 5 gives a discussion about the proposed model and discusses some implementation decisions. It provides a review of the general results and contributions of this thesis as well as ideas for future work.

Chapter 2

State of the Art

Information technologies play an important role in supporting solutions which can increase productivity, quality of service, and at the same time keep low operation costs. The healthcare area is certainly not an exception, thus raising the emergence of a new area – medical informatics. Research into this topic focuses mainly on how to solve medical problems and challenges by using information technologies. One such challenge is digital medical records. This subject has been a hot research topic focusing on several issues and as a consequence new types of records have been developed as well as new standards and solutions for record integration and sharing.

Previous research on digital medical records targeted requirements that have been evolving because of the dynamic scenario of healthcare provisioning combined with changes in how people live and work. In this context, state of the art analysis is guided by a methodology, illustrated in figure 2.1. The first step goes through understanding what digital medical records are, their types, and standards for improving awareness on this area. Afterwards, identifying unresolved challenges, related to patient mobility and the change in healthcare.

Those challenges precipitate the need of understanding how records can be integrated. Then, briefly analyzing the regulatory framework to understand the requirements trains of medical data manipulation to perceive how the access can be secured.

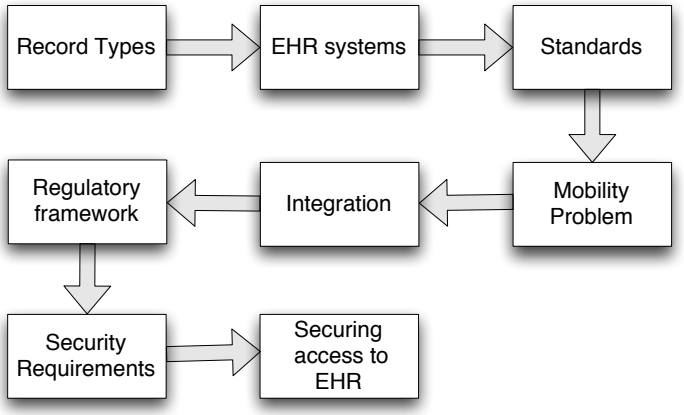

Figure 2.1: State of Art Structure

2.1 Record Types

Normal medical procedures aim at recognizing a disease or a health dilemma, based on a set of symptoms and signs. To facilitate the process, the physician will try to build the diagnostic based on physical signs, medical tests, and the patient's historical data.

Test results, such as blood pressure, medical imaging, electrocardiogram, among others, is critical information that contributes to the patient's medical history – or health record, forming a valuable insight for future diagnosis.

The health record is typically stored at a specific healthcare provider, usually the caregiver, or the laboratory which is responsible for the exams. Due to this, there is a tendency to scatter information, thus creating difficulties in building and accessing a full, integrated, record. The dispersion becomes even more critical as the information gets more fragmented as a result from citizens' mobility, the increase of the number of actors and of the type of procedures.

Many of these records are still paper based [16], with the associated problems of illegibility, unavailability, sheer physical volume (over time), difficult transferability, and integration between providers and institutions, besides the necessity to record the same data often (duplication) on different documents [17, 18, 19]. Paper-based records also require staff for routing, archiving and maintenance. Different authors state that the paper-based patient record is therefore reaching its limits [20]. Digital versions of those records can bring many benefits, as they can solve some of the above mentioned problems and can enable new ways of dealing with medical information [16].

The forthcoming sections focus on some digital health records, namely EMR, EHR and PHR.

2.1.1 Electronic Medical Record

The EMR is a detailed record that contains all the information generated during the patient stay at a Care Delivery Organization (CDO). Professionals within the CDO create, maintain and use the EMR. The core components of the EMR are the Clinical Data Repository (CDR), Clinical Decision Support System (CDSS), controlled medical vocabulary (CMV), Computerized Provider Order Entry (CPOE), pharmacy management system and electronic Medication Administration Record (eMAR) [1] (Figure 2.2).

The record is owned by the CDO. The creation and use of information is done through systems that are deployed in hospitals, health systems and clinics. In these records the patient may access to some information if the CDO authorizes it. The interoperability is not addressed since each EMR only contains information from a single CDO.

2.1.2 Electronic Health Record

The EHR can be described as a longitudinal storage of patient health information generated by any encounters at any care delivery setting [21]. This information may include several types of information such as patient demographics, progress notes, problems, medications, vital signs, medical history, immunizations, laboratory data, and radiology reports. The EHR can generate a complete record of a clinical patient encounter, as well as support other care-related activities directly or indirectly via external interfaces.

EHR have two main scopes: the core EHR and the extended EHR. The core EHR is defined by ISO/TS 18308 and aims at providing a record that supports present and future

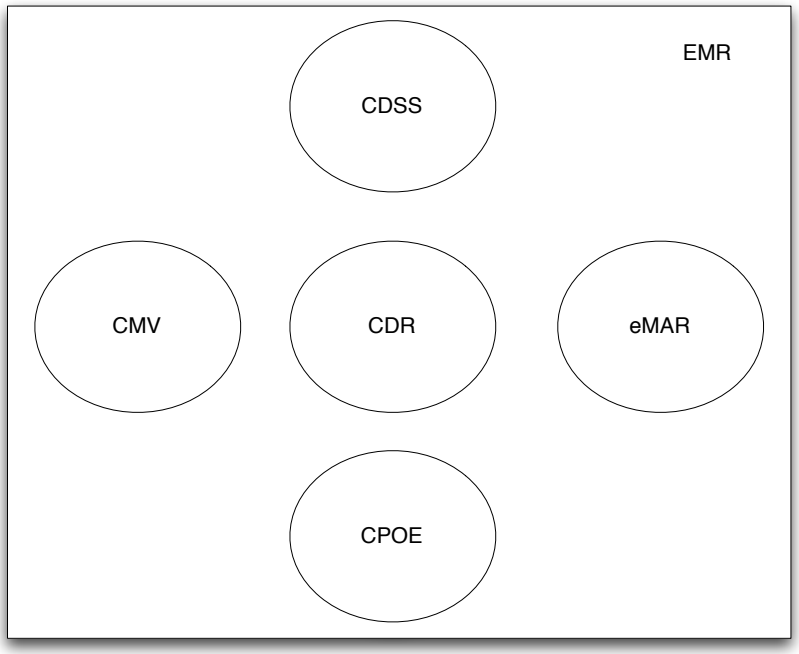

Figure 2.2: EMR core components

healthcare provided by clinicians, where the primary beneficiaries are the clinicians and the subject of care. Basically, it is a container that stores patient-centred clinical information.

The extend EHR makes use of information defined by secondary users for the purposes of billing, policy, and planning, statistical analysis, among others. It supports functions such as patient administration, scheduling, billing, decision support, access control and policy management, order management, guidelines, terminology, population health recording, querying and analysis, health professional service recording, querying and analysis, business operations recording, and resource allocation. The extended EHR can be used to associate knowledge extracted from the patient clinical information with other information from external sources, thus, enabling the creation of decision support or/and inference systems.

As already stated, other types of records exist. One that can generate some confusion with the EMR is the EHR. The latter can be described as a longitudinal storage of patient health information generated by encounters in any care delivery setting [21]. Despite of some controversy and confusion, both the EMR and EHR have discrepancies [22]. The second contain information usually comprised in the CDR of diverse EMRs, since EHR cover diverse health-care occurrences in multiple CDO [22]. The differentiation between EMR and EHR is shown in Table 2.1.

The differentiation between EMR and EHR can be analyzed in terms of definition, owner of the information, usage scope, rights of the patient and interoperability with others. The EMR has CDO scope, while the EHR can combine a subset of information from different CDOs. The owner of the information of the EMR is the CDO, contrasting with the patient or stakeholder of EHR. The range of use is the CDO for the EMR and on the other hand the EHR can have a community, region, or country scope. This has a direct effect in interoperability, since the EMR does not have functionalities for sharing because EMR was designed to be used by only one CDO. By contrast, EHR enables information sharing between multiple

Table 2.1: Differentiation between EMR and EHR [1]

Record Type	EMR	EHR
Definition	The legal record of clinical services for a patient within a CDO.	A subset from one or more CDOs where the patient received clinical services.
Owner	Owned by the CDO	Owned by the patient or stakeholder
Consumer and Usage Scope	EMR systems are supplied by enterprise and installed by hospitals, health systems, clinics, etc.	EHR systems are run by community, state, or regional emergence, or national wide emergence organizations.
Right of patient	Patient can gain access to some EMR information once authorized by the EMR owner.	Patients are provided with interactive access as well as the ability to append information.
Interoperability with other CDOs	Each EMR contains patient's encounter in a single CDO. It does not contain other CDO encounter data.	Sharing information among multiple CDOs, connected by National Health Information Network.

CDOs connected by a national health network. Also, patient rights are more restrictive in EMR than in EHR.

The CDR is organized in a service-centric view, in opposition to the EHR, which is patient-centric. The CDR normally does not support version control, privacy, and other medical and legal characteristics. As such it can be used as a source system for the EHR, since information from a specific patient can be queried from a CDR of an EMR for populating the patient-centric EHR.

The standard ISO/TC 215 [23] defines and makes the contextualization of the different types of EHR records and systems. A basic definition was created to explain the concepts more easily. Then, the definition was gradually specialized to aggregate common characteristics that enables defining other types of records (Figure 2.3).

The Basic-Generic EHR *"is a repository of information regarding the health status of a subject of care, in computer processable form"* [23]. It is a record that stores health information corresponding to a subject in a form which can be processed informatically. The subject of care represents a patient, client, or consumer, depending on the context. It usually identifies a single individual but can also identify a group.

One of the most important characteristics of the EHR is the ability to share information between authorized users. Information sharing is useful if the meaning is not lost. In other words, records and systems have to be interoperable, fostering an EHR model independent of the persistence technology. Moreover, physical database structure should be independent of the applications which manipulate the data. These are fundamental factors to enable a *future proof* EHR, supporting the dynamic environment of healthcare records.

Interoperability can be contemplated at two levels: functional and semantic. The first includes the ability of two or more systems to exchange information in a human readable form, and depends on a standardized EHR reference model,as well as standardized service

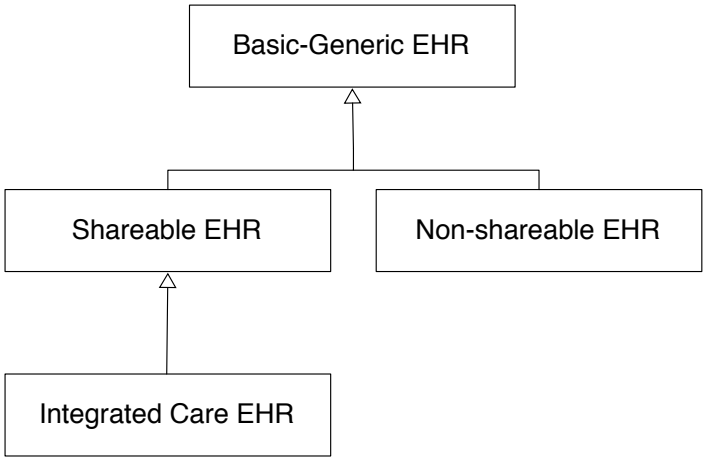

Figure 2.3: Types of EHRs
[23]

interfaces models. The latter includes the ability of the destination system to autonomously process information. Usually this depends on several layers: terminologies, on the content of archetypes[1] and on templates[2] used between the receiver and the sender. Standards are also needed for group domain-specific concept models and terminology use in EHR.

Shareable EHR enables sharing information between different systems and can occur at three levels:

- among different clinical disciplines or other users, all of whom may be using the same application, requiring different or ad hoc organization of EHRs – **level one**;
- among different applications at a single EHR node (i.e. at a particular location where the EHR is stored and maintained) – **level two**;
- across different EHR nodes (i.e. across different EHR locations and/or different EHR systems) – **level three**.

Shareable EHRs on level one and two usually contain detailed information required for patient care at a single location – the information is created and maintained in a local EHRs system. The information normally includes a summary such as a problems list, allergies, past medical history, family history, current medication, among others. At level three, sharing is characterized by supporting integrated care of patients between different care delivery organizations and is typically called an ICEHR. This will be discussed in section 2.1.3.

The definition of non-shareable EHR is done by exclusion. In short, it is an EHR that does not support information sharing between EHRs. Mostly, because of EHRs that are based on proprietary information models within EHR systems that are not interoperable between different EHR systems. This type of EHR is usually bound to a specific system and database. Those are the most implemented systems in all health areas, thus making integration more difficult.

[1] **Archetype** – a computable expression of a domain-level concept as structured constraint statements, based on some reference information model.

[2] **Template** – a directly, locally usable data creation/validation artefact which is semantically a constraint/choice of archetypes and which will often correspond to a whole form or screen.

Information contained in the EHR is produced by healthcare professionals and maintained by health-care providers, following four types of models: the fully federated, the federated, the service orientated, and the integrated [24]. Moreover, the deployment in each country/region or federation uses different approaches under different regulatory frameworks. With the lack of well defined standards, interoperability becomes difficult [25].

The EHR is mainly devoted to facilitate the work and information flow between different departments of an institution or federation. They also try to manage administrative information associated with admission, discharge and payments [25]. Typically this approach excludes any patient intervention, even on the requirement analysis. It is a solution to cope with healthcare professionals needs, inside well-defined groups of actors. The sharing of patients' clinical information is supported by agreements between them. Patients start to gain access to their EHR, but only to very limited views in specific areas for reading, without the possibility for collaboration, and without access control privileges.

2.1.3 Integrated Care EHR

The ICEHR is defined as a *"repository of information regarding the health status of a subject of care in computer processable form, stored and transmitted securely, and accessible by multiple authorized users. It has a standardized or commonly agreed logical information model which is independent of EHR systems. Its primary purpose is the support of continuing, efficient and quality integrated health care and it contains information which is retrospective, concurrent, and prospective"* [23].

This type of EHR urges a standardized logical information model, persistence of information, and solutions to provide security and privacy of the records.

A minimum level of functional interoperability is needed to share information, but semantic interoperability would also increase the value of the sharing. The standardization of clinical terminology as well as the one from other domains, archetypes and templates can be a solution for achieving semantic interoperability. Various logical information models are being developed by organizations such as International Organization for Standardization (ISO), European Committee for Standardization (CEN), Health Level Seven (HL7) and the OpenEHR project.

The information contained in this type of records is lifetime information and as such demands long-time persistence solutions, which include semantics for information storage, version control, and rules regarding the modification and deletion of information. This characteristic distinguishes the EHR from the messaging paradigm (e.g. HL7), which has no persistence, although solutions can be used to extract the information from the message and save it in some persistent artifact or in the EHR. Another important consideration to be made concerns how the information can be manipulated, especially in case of deletion and update. From the medical and legal point of view, the information saved should not be deletable. Instead, if the information is flawed, a new version should be created and used in future accesses to the EHR. The old version will only remain available for legal purposes. However, it should be possible to completely delete information, as, in some jurisdictions, the patient can request the deletion of erroneous information. There may also exist a specific time lapse after which the entire health record may be permanently deleted. Each ICEHR should be as complete as the local circumstance allows to and it should have the capability for contributing its information to other larger-scale ICEHR systems.

Last, but not the least, the ICEHR has security and privacy concerns. Most jurisdictions

consider privacy and security essential to medicolegal integrity, community trust and acceptance of EHRs. The concerns are naturally related to information security when stored and transmitted. Part of the concerns is also the regulation of access by multiple authorized users. This is considered critical. EHR requires access control mechanisms, allowing access by the subject of care (where suitable and allowed by local laws and policies) and other authorized users (e.g. the subject's agent such as a parent of a child subject) as well as healthcare delivery professionals.

2.1.4 Electronic Health Record Systems

EHR Systems are responsible for manipulating health records. Some authors only include information technology components while other extend them to people, procedures, and rules.

Systems are characterized according to record manipulation, creation, store and usage rules. They can be Local-EHR Systems and Shared-EHR Systems, regardless of the EHR type. In other words, it is possible to have a Local-EHR System using a Shareable EHR.

A Local-EHR system is mainly oriented for supporting the patient healthcare provisioning inside a single organization. It can deal with information from external sources, such as diagnostic results and referrals. The access to this system is usually restricted to authorized users from within the same organization and, in some systems, to the subject of care. This system has to deal with different type of actors, such as primary care clinicians, specialists, complementary practitioners, and other healthcare professionals.

A Shared-EHR System enables sharing information among different organizations. The ultimate goal is to enable integrated healthcare within a community of providers. It also supports the sharing of extracts and integrated workflow between organizations [23].

As discussed previously, the Integrated Care EHR is one of the most prominent and desirable to achieve. Systems that cope with most of the requirements to deal with this type of record are the Federated ICEHR and the Consolidated ICEHR. Both models are based on a Shared-EHR System.

The Federated ICEHR model creates the view in real time. It can be considered a virtual EHR, since it creates a view of the information, at request time, assembling on-the-fly the information from distributed EHR nodes. The system usually depends on a service that gathers the links correspondent to patients' information, following an EHR directory service. The system is a meta-EHR witch contains no personal health information but rather a set of links to distributed EHR nodes for particular subjects [23]. It possesses several implementation challenges as well as performance issues, related to difficulties for achieving efficient query and retrieve methods. Additionally, security requirements may also demand complex security models.

The Consolidated ICEHR model assembles the information generated in the creation or actualization steps. Data is consolidated through insertion or update, instead of creating a view in real time when info is requested. When the information is created, it can be pushed from a Local-EHR System to the ICEHR system or it can be created directly in the ICEHR. This model minimizes performance issues, reduces the attack surface and because it is a less intrusive solution, it facilitates the creation of access control and other security mechanisms, when compared to a federated scenario. The consolidated model is currently being adopted with regional and national Shareable EHR. Some projects are running in Australia, Brazil, Canada, and in the United Kingdom [23].

2.1.5 Personal Health Record

During the last decade citizens became more informed on healthcare subjects. They demand to have a more active role in their own healthcare, willing to contribute in the creation and update of information in their record. A response to those requirements was a new type of record – the PHR – thus enabling the subject of care to have the control of its own record. In this solution the subject of care creates most of the information, if not all, on the record.

The PHR can be described as a lifelong tool for managing relevant health information of an individual [26]. It promotes personal information maintenance and may be used in a broad scope or in more specific scenarios, such as chronic disease management. The PHR is owned, managed and shared by the individual or a legal proxy. Usually it is a self-contained EHR, maintained and controlled by the subject of care, although it can also be maintained by a service provider. Alternatively, it can be a component of an ICEHR, maintained by a healthcare provider and controlled, partially at least, by the subject of care.

Some authors advocate that the PHR can have the same record architecture and standard information model of a EHR and still cope with patient empowerment [23]. Using a standard information model for all types of EHRs and ICEHR promotes information sharing between all systems when necessary and controlled by the patient.

The most relevant types of PHR are the standalone and the web-based. The standalone normally makes use of some external storage device [27], and information is typically created by the patient. Ownership works as access control, usually combined with different privileges that the patient chooses to associate with each record. Web-based PHRs are generally stored in a central repository and define a set of core features that may be extended by third-party services. However, the use of external services requires previous approval by the providers of the web-based PHRs. The most prominent web-based PHR, to this date, are Google Health, Microsoft HealthVault and Dossia. However, Google Health was, in the meantime, discontinued.

Most PHRs use proprietary data formats which hinder the collaboration between different actors. Google Health and Microsoft HealthVault make use of standard formats or a subset of a standard as their record format. The standard formats most used in this area will be discussed in section 2.2.

Table 2.2 presents the main differences between EHR and PHR. The analysis is made contemplating the following properties: the guardian of the information, data creator, who defines the access control and who provides the system.

Table 2.2: EHR vs PHR

Record Type	EHR	PHR
Guardian	Providers	Patient or a service on his behalf
Creation of data	Medical Staff	Patient or exported by services
Sharing	Institutional agreements	Patient choice
Access Control	Provider	Controlled by the patient
System Provider	Providers	External Service Provider

In brief it can be seen that the healthcare provider controls and owns the patient EHR, in

contrast with the patient empowerment supported by the PHR. It is important to highlight that the system provider of the PHR can be external to institutions or federations.

2.1.6 Discussion

The relation between EMRs, EHRs, and PHRs is described in Figure 2.4. There is an overlap, because some information exists also in other records.

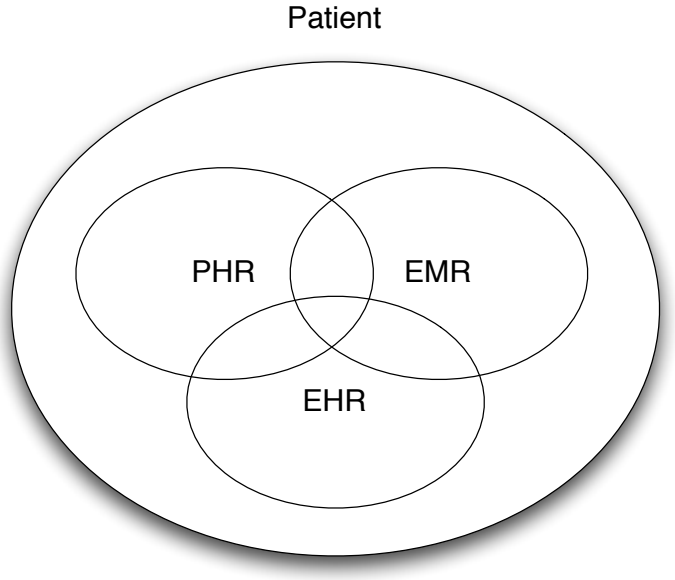

Figure 2.4: Relationship of PHR, EMR and EHR [1]

As discussed previously, EMR can feed information from its clinical repository to an EHR. Moreover, different clinical repositories can be aggregated in an EHR. The owner of the information changes from the institution that has the EMR to the patient or a stakeholder. The PHR can have information that is complementary to the EHR/EMR and similar information that resides in those records.

In the context of mobility, the EMR is not eligible, since it is orientated for a single CDO. The EHR and PHR subsist as alternatives for mobility support. The differences leave an open challenge since the EHR has the trust of the medical staff, but record sharing is difficult (Table 2.2). It also restricts patient's freedom of choice, since he is dependent on agreements between providers for gaining or giving access to his medical information. In this scenario, the patient appears as a passive actor, as he cannot contribute to his record and cannot control the access to his medical information.

The PHR, on the other hand, seams to cope better with most of the mobility requirements. It enables sharing between different actors, regardless of their location and agreements. It further depends on patient approval. It also solves the dilemma of the infrastructure cost, as the patient can choose a PHR provider. One drawback is the trust on the integrity of the clinical information by the clinical staff.

Based on this analysis, a new type of record is proposed – the Hybrid Electronic Health Record which can take advantage of the best characteristics of the EHR and PHR for mobility and open collaboration scenarios (section 3.1).

2.2 Standards

Standardization is essential to enable communication between different systems. Several organizations, such as the European and American committees, country and the World Health Organization, are making efforts to achieve this goal. These initiatives are pushing forward research, and gathering evidence about interoperability barriers between standards.

Some efforts were made in EHR requirement analysis and specification according to the SO TC 215 Health Informatics and the CEN ENV/EN 13606 EHRcom standards. Complementary efforts were divided in two main areas: the communication standard and the document standard [2, 28] (Figure 2.5). The former focus on how systems can communicate with each other and the latter delineates how information is stored to ensure a correct interpretation by other systems.

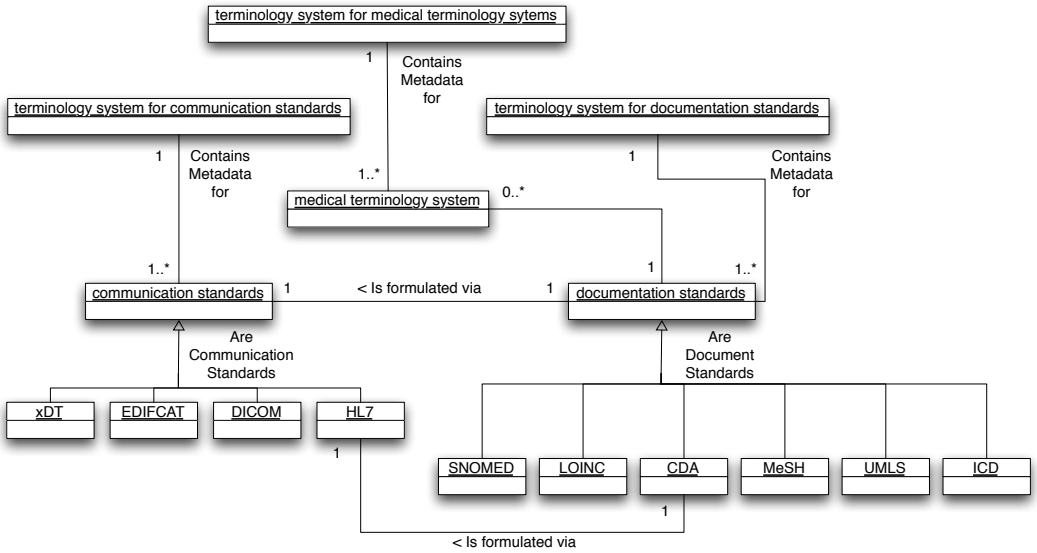

Figure 2.5: Medical communication and documentation standards [2]

Interoperability is one of the big challenges in the EHR area, thus considered fundamental to enable sharing between healthcare providers [29]. The functional interoperability, addressed by communication standards, is the ability of different systems to communicate among them. Semantic interoperability aims at ensuring that medical significance is not lost or disrupted when sharing the record with other system. Both make use of terminology systems oriented for each area, to enable a common domain terminology for the standards.

In the medical area, some efforts have been done regarding communication standards. HL7, Digital Imaging and Communication in Medicine (DICOM), Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT) and the xDT-standards are some examples of that.

HL7 is an international, vendor-independent, communications standard in healthcare for information exchange between systems and institutions [30]. HL7 was created in 1987 as a consortium founded by a group of healthcare providers (Figure 2.6). They created the first specification after one year, and two year later version two, following a pragmatic path to enable communication between systems and institutions based on message exchange [31].

The core concept of HL7 version two is based on an external event, a so-called trigger

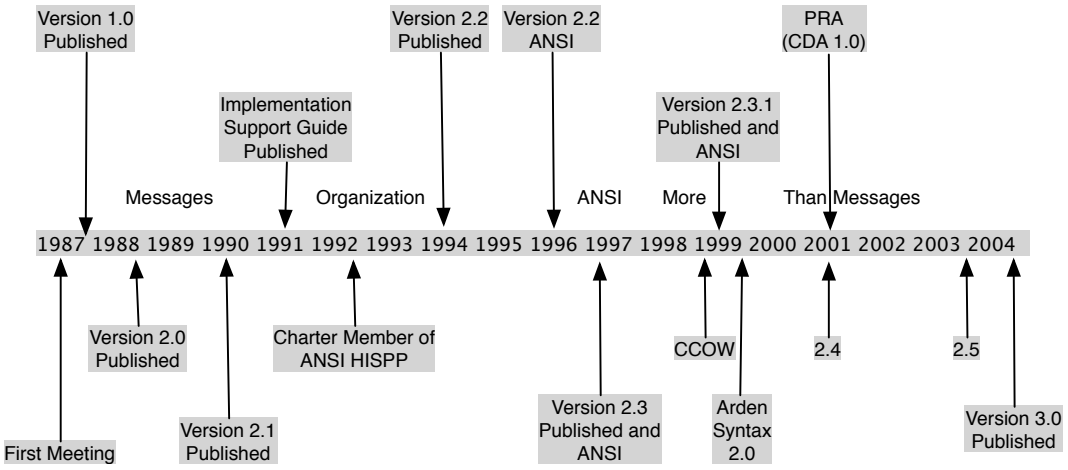

Figure 2.6: HL7 Timeline [3]

event. When it occurs it is recognized by a healthcare computer application [31] and a specific message is generated. HL7 does not specify the communication protocol, dealing only with triggers and messaging. Messages in version two are delimited American Standard Code for Information Interchange (ASCII) strings, divided into segments and into fields within the segments. Normally, the information in each segment is related to a particular concept or entity in healthcare domain.

Version three uses Extensible Markup Language (XML), thus supporting structured data. This facilitates the integration of data, since libraries for XML manipulation are available. HL7 operates at application level of the ISO/OSI reference model. The Clinical Document Architecture (CDA) expands the HL7 standard with the description of the structure and the contents of clinical documents, based on XML format [2].

DICOM is an open standard for the exchange of images and additional meta-information (patient's name, admission date, device parameters, attending physician,...) in healthcare. The standard describes data fields, functions for network services as well as syntax and semantics for the commands and messages. DICOM can store images in TIFF or JPEG format, enabling electronic archiving of images in all medical information systems [2].

One important joint effort between the DICOM and ISO is the Web Access to DICOM Persistent Objects (WADO) standard. The latter defines a web-based service that can be used to retrieve DICOM objects, such as images, waveforms and reports. The access is made through Hypertext Transfer Protocol (HTTP) or HyperText Transfer Protocol over SSL (HTTPS) [28]. It does not support a query mechanism, so the client has to directly specify the DICOM object to be retrieved: the unique identifier to the study, the series, and the instance. The server answer can encapsulate the DICOM object in the HTTP/HTTPS protocol, enabling the integration with HyperText Markup Language (HTML) pages or XML documents [32].

The query is made by a HTTP request using the GET command. An example of retrieving a region of DICOM image converted in JPEG2000, with annotation burned into the image containing patient name and technical information is depicted on Figure 2.7. It also enables other options, e.g., simple retrieve of DICOM objects in JPEG, DICOM structure report in HTML, retrieved as a DICOM mime type, without identification, ... [4].

WADO is normally used by web-based EHR to reference DICOM images, send them by

```
https://aspradio/imageaccess.js?requestType=WADO
&studyUID=1.2.250.1.59.40211.12345678.678910
&seriesUID=1.2.250.1.59.40211.789001276.14556172.67789
&objectUID=1.2.250.1.59.40211.2678810.87991027.899772.2
&contentType=image%2Fjp2;level=1,image%2Fjpeg;q=0.5
&annotation=patient,technique
&columns=400
&rows=300
&region=0.3,0.4,0.5,0.5
&windowCenter=-1000
&windowWidth=2500
```

Figure 2.7: WADO GET [4]

email or for image distribution and sharing between different EHR systems, since it can enable access to DICOM images through a simple web browser.

EDIFACT standardizes formats for electronic exchange of administrative data, typically including orders, calculations and payment orders. EDIFACT was not specifically produced for the healthcare area, although it is being used globally by business partners of different areas to exchange accounting-relevant data [2].

The xDT-standard is used in Germany in the general practitioners sector. It specifies different data formats to simplify the communication, and data exchange between healthcare providers and health insurance companies. Some efforts are being made to harmonize the communication standards through XML, taking advantage of the wide availability of programming libraries and of the openness of the XML, which can also be read and understood by users [2].

The Logical Observation Identifiers Names and Codes (LOINC) is used to correctly identify diagnosis, using multidimensional terminology system for clinical laboratories. It allows detailed descriptions of medical circumstances for almost any clinical problem to be solved automatically [2].

The International Classification of Diseases (ICD) is used for illnesses classification and related health issues [2].

The Systemized Nomenclature of Medicine (SNOMED) is a nomenclature that combines terms that are based on determinate concept orders. It covers the arrangement of unified terminology expressions in the medical field, supporting diverse languages [2].

The thesaurus Medical Subject Headings (MeSH) is an extension of a nomenclature. It enables indexing international publications, making use of a metathesaurus with semantic and linguistic information. The Unified Medical Language System (UMLS)) is a metathesaurus which tries to integrate all important medical terms in only one term, and to represent all possible term relations accordingly [2].

Although trying to solve interoperability problems, all these standards can pose compatibility issues in documents if using different terminologies. To achieve harmonization between the standards and thus enabling interoperability, two main groups exist: the industry that makes use of proprietary standards like HL7 and DICOM and university research, such as the CEN/TC 251 – European Standardization of Health Informatics.

In order to illustrate how harmonization can be achieved, CEN/TC 215 Health Informatics and CEN ENV/EN 13606 EHRcom, HL7/CDA documentation format, and OpenEHR can be discussed. The initiative Integrating the Healthcare Enterprise (IHE) will also be studied.

2.2.1 CEN TC 215 Health Informatics and CEN ENV/EN 13606 EHRcom

The pre-standard CEN/TC 13606 EHRcom "Electronic Healthcare Record Communication" is a message based standard for the exchange of Electronic Health Records (EHR). It defines an EHR information model, extended from CEN ENV 12265, with a list of machine readable domain terms which can be used for structuring the EHR. It also describes methods for specifying the rules for EHR sharing, allowing the creation of request and response methods to support communication between EHR systems. The CEN ENV/EN 13606 does not attempt to specify a full EHR. It only describes interfaces relevant for communication between the different EHR systems [28].

Different countries tried to implement systems using this standard, but none of them implemented the full standard. Even semi-implementations demonstrated some weaknesses. As the single-level modelling approached, it augmented the complexity of the information model. It has many options and a high level of abstraction in the model, which makes it very difficult both to understand and to implement [28].

In 2001, CEN TC 215 revised the CEN ENV/EN 13606 targeting a full European Standard, using the previous experience and adoption of the OpenEHR (on subsection 2.2.3) archetype methodology. The standards has five parts [29]:

- Reference Model;
- Archetype Interchange Specifications;
- Reference Archetypes and Term Lists;
- Security Features;
- Exchange Models.

Of the previous five parts, only the Reference Model is stable, while others are still under development. The Reference Model is a generic model for manipulating a patient EHR that enables interoperable communication between different systems, composed by different packages [33]:

- Extract package;
- Demographics package;
- Terminology package;
- Data Type package;
- Access Control package;
- Message package.

The Extract Package is the root level of the Reference Model and defines the data structures for the EHR content (Table 2.3). The Demographics package deals with minimal data to define persons, software agents, devices and organizations which are referenced in the EHR. The Terminology package defines the terms used within the EHR and the primitive data types and values are defined in the Data Type package. The Access Control package defines

Table 2.3: EHRcom blocks [5]

Block	Description
EHR	The electronic healthcare record for one person
Folders	High-level organization of the EHR, e.g. per episode, per clinical speciality
Compositions	A clinical care session, encounter or document, e.g. test result, letter
Sections	Clinical headings reflecting the workflow and consultation process
Entries	Clinical "statement" about Observations, Evaluations and Instructions
Clusters	Nested multi-part data structures (tables and interval time series) e.g. audiogram
Elements	Leaf nodes with single data values, e.g. reason for encounter, body weight
Data Values	Data types for instance values, e.g. coded terms, measurements with units

a representation of access policies for the EHR, and, finally, the Message package will deal with attributes that will be sent via message in a serialized form after a request process.

The top level is a directory which corresponds to one patient. It may have nested folders that correspond to an episode or a clinic speciality. Those folders can have zero or more compositions. Each composition can roughly correspond to one clinical document. Each document has section headers and entries which can be simple elements or a cluster of elements. Each element has a single value, or a single data type. The content of the EHR is always appended or replaced as a composition [29].

The second important building block is the archetype concept which uses a two-level methodology to model the EHR structure. The first level uses a generic reference model for the healthcare domain, which contains few classes that will be stable over time. On the second level, the healthcare and application specific concepts, e.g., blood pressure, and lab results, among others, are modelled using archetypes. The archetypes are used to describe clinical concepts, separating the domain expert knowledge from the implementation [5]. This approach was put forward by the OpenEHR, which will be studied in subsection 2.2.3.

2.2.2 HL7/CDA

The HL7 Clinical Document Architecture is a document markup standard. It was developed by the HL7 to deal with persistence. It was first designated as Patient Record Architecture (PRA)) and comprises three levels (Table 2.4).

CDA documents are encoded in XML, deriving their meaning from the HL7 Reference Information Model, making use of HL7 Version three Data Types. It has three different levels of granularity. Each level appends more markup to clinical documents, while the content remains constant at all levels. Level One focuses on the content of narrative documents, such as parties, roles, dates and times, places and structural organization headings. The document is divided in two parts both based on the HL7 data types: the CDA Header and the CDA Body. The header is derived from the Reference Information Model. It defines, without ambiguity, the semantics of each entry in the document. The body contains the clinical

Table 2.4: Hierarchy of CDA Level in Release One and Two [5]

CDA Release One	CDA Release Two
CDA Level One	The unconstrained CDA specification
CDA Level Two	The CDA specification with section-level templates applied
CDA Level Three	The CDA specification with entry-level templates applied

document content. It can be unstructured text or nested in sections, paragraphs, lists and tables using the structured markup language. Level One does not provide semantics, enabling only interoperability for human readable content. It describes only a document analogous to HTML, with a standardized header that contains additional information about the document itself.

On Level Two, the specification brings a fine-grained observation and instructions within each heading using a set of Reference Information Model classes. It enables constraining the structure and the content of the document through templates, hence, boosting interoperability, since systems are able to understand the receiving data. The CDA has its base architecture defined and provides different granularity of constrains. This approach facilitates the migration from free text to more structured CDA documents [5].

A complete structured document is only available at Level Three. Each information entity is specified using a unique code, enabling machine document processing (Figure 2.8. In order to enable automated document process, the blocks are encoded in XML, containing coded elements.

The information on the section enables to apprehend that it was coded using the LOINC and the observation element was coded using the SNOMED to describe a *Skin finding*. As the codes are unique, they can be used for machine process. Using document-level, section-level, and entry-level templates enables to constrain the generic CDA specification [28].

The CDA standard does not specify services or protocols that are used for document exchange. In a HL7 message the CDA document is just a multimedia object, which can be exchanged as a Multipurpose Internet Mail Extensions (MIME) package [28].

Some projects are using CDA as a format for clinical documents, mostly used in clinical information systems that already use the HL7 standard. The CDA is not an EHR standard since it define only parts of an EHR architecture. It is an important part of an EHR and is therefore being harmonized with equivalent structures in EN 13606 and OpenEHR. CDA is a subset of the EN 13606 Reference Model and the EN 13606 will be compliant with CDA Release Two [28].

2.2.3 OpenEHR

The OpenEHR finds its genesis in the Good European Health Record (GEHR) dating back to 1992. It started as an European research project with strong participation of Australia. Nowadays the project is maintained by a nonprofit organization – the OpenEHR Foundation. It aims at the creation of a non-proprietary universal independent application and health record, supporting accurate and safe health information exchange. According to the types of EHR, it corresponds to the Integrated Care EHR. In practical terms it is:

```
<section>
    <code code="8709-8" codeSystem="2.16.840.1.113883.6.1" codeSystemName="LOINC"/>
    <title>Skin Exam</title>
        <text>Erythematous rash, palmar surface, left index finger.
            <renderMultiMedia referencedObject="MM2"/>
        </text>
    <entry>
        <Observation>
            <code code="106076001" codeSystem="2.16.840.1.113883.6.96"
                codeSystemName="SNOMED CT" displayName="Skin finding"/>
            <value xsi:type="CD" code="271807003"
                codeSystem="2.16.840.1.113883.6.96"
                codeSystemName="SNOMED CT" displayName="Rash"/>
            <targetSiteCode code="48856004"
                codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED CT"
                displayName="Skin of palmer surface of index finger">
                <qualifier>
                    <name code="78615007" codeSystem="2.16.840.1.113883.6.96"
                        codeSystemName="SNOMED CT" displayName="with laterality"/>
                    <value code="7771000" codeSystem="2.16.840.1.113883.6.96"
                        codeSystemName="SNOMED CT" displayName="left"/>
                </qualifier>
            </targetSiteCode>
            <entryRelationship typeCode="SPRT">
                <RegionOfInterest MMID="MM2">
                    <id root="10.23.4567.4489"/>
                    <code code="ELLIPSE"/> <value> 3 1 3 7 2 4 4 4</value>
                    <entryRelationship typeCode="SUBJ">
                        <ObservationMedia> <id root="10.23.4567.345"/>
                            <value xsi:type="ED" mediaType="image/jpeg">
                            <reference value="lefthand.jpeg"/> </value>
                        </ObservationMedia>
                    </entryRelationship>
                </RegionOfInterest>
            </entryRelationship>
        </Observation>
    </entry>
</section>
```

Figure 2.8: Part of CDA Level Three Document Body [5]

- patient-centric: each EHR is related to one subject of care (not an episode of care at one institution);
- longitudinal: it is a long-term record of care, possibly from birth to death;
- comprehensive: includes all records of care events from all types of careers;
- prospective: not only historical events are records, also decisional and prospective information as plans, goals, orders and evaluations.

OpenEHR is modeled according to a two-level paradigm (Figure 2.9). The first level (Reference Model) defines a reference information model, and the second level define archetypes

and templates, formal definitions of clinical content [6].

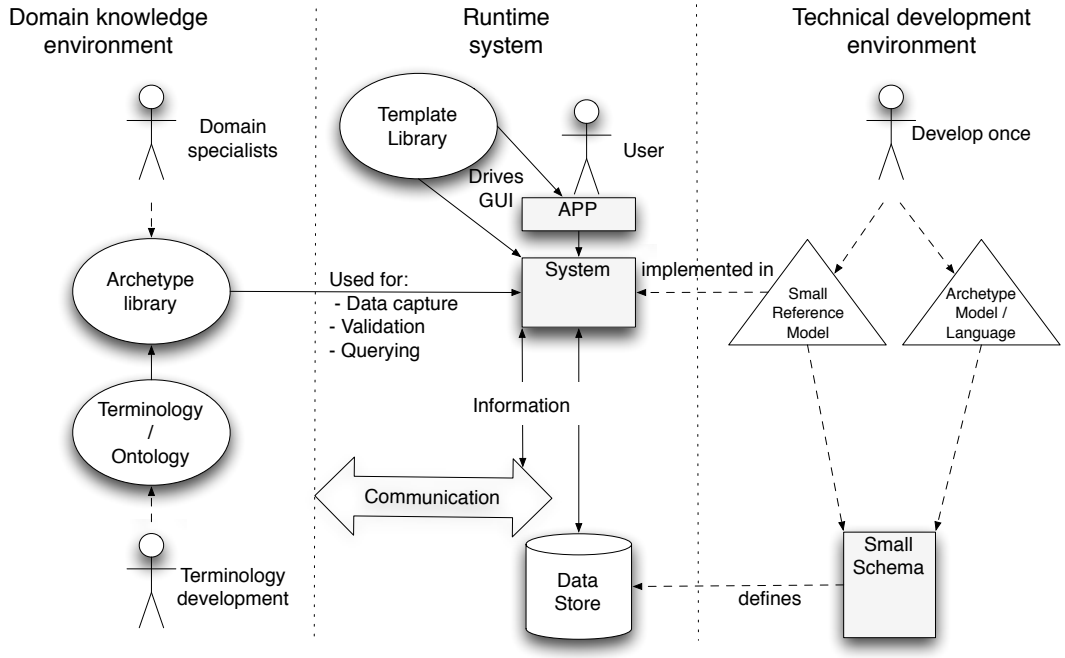

Figure 2.9: Two-level Software Engineering [6]

Only the first level is implemented in software, enabling a small and easy to maintain the system, in contrast with a single-level system. This approach also allows the systems to be future proof as it can more easily adapt to changes. This is possible because the systems are built to consume archetypes and templates, created upon the Reference Model, separating the clinical modelling, where the domain experts work, from the software developer.

Moreover, using this approach, archetypes can be used as semantic gateways to terminologies, classifications and even computerized clinical guideline services, thus, enabling to separate from the system in the form of archetypes and templates (Figure 2.10).

All interaction with data is achieved through archetypes and templates. An archetype is a conceptual component that allows the standardization of data by mapping each medical event into pre-agreed individual fields of information (entries) forming a composition. A template serves as a messaging standardization structure and is closely related to screen forms. It aggregates one or usually more archetypes.

Archetypes in OpenEHR are formalized using Archetype Object Model (AOM), that, when in memory, define the semantics of the archetype. The OpenEHR makes use of the Archetype Definition Language (ADL) to define the archetypes, allowing serialization through XML [34]. For human view and screen rendering different formats like HTML and RTF are used [6].

The description of an archetype has three parts: descriptive data, constraint rules and ontological definitions. Descriptive data has the archetype ID, a description of the modelled clinical concept, the version and purpose. The constraint rules specify the structure, cardinality and content to be valid in the OpenEHR Reference Model. The ontological definitions describe the vocabulary in machine readable codes, sometimes using terminology that can be applied in some entries on archetypes instances [8].

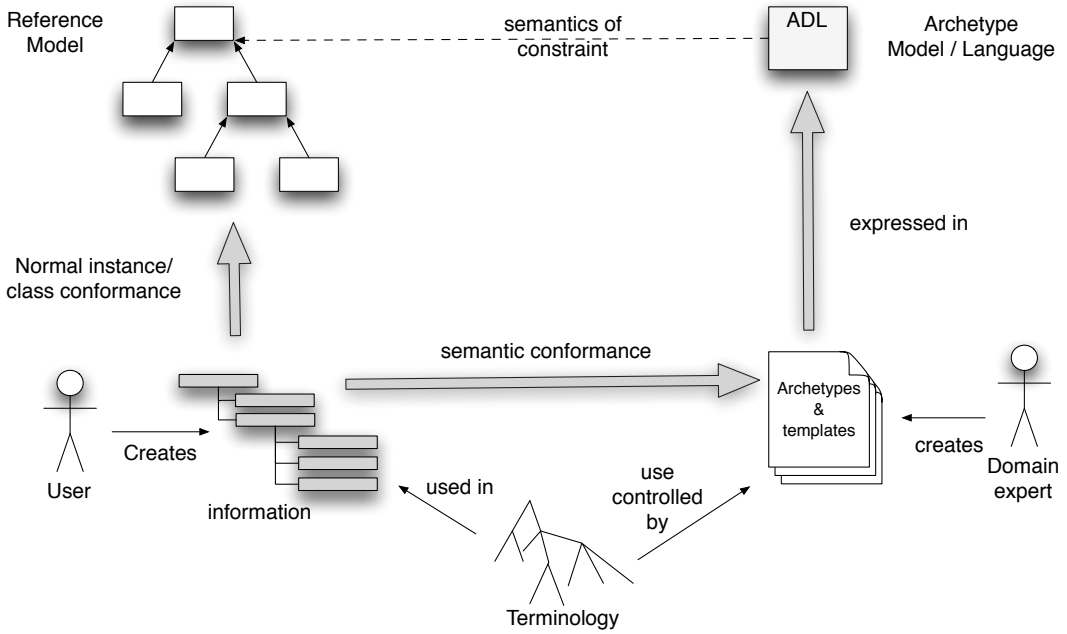

Figure 2.10: Archetype Meta-architecutre [6]

A repository of heavily reviewed archetypes is available to the community fostering the collaboration and future use of the OpenEHR. Archetypes with different scopes exist, some are developed for specific countries or special use and others are even globally suitable. They are available in the OpenEHR website at `http://www.openehr.org/knowledge/`. It enables searching and downloading archetypes in ADL or XML format, as well as the creation and revision of archetypes by all community members.

A query language is also available – Archetype Query Language (AQL), for enabling querying instantiated archetypes. It enables access to the nodes using an analogous approach of an XPath query on a XML document [35].

A minimal OpenEHR System consists in an EHR repository, an archetype repository, a demographic repository, and other times a terminology service (Figure 2.11).

OpenEHR completely separates the EHR from demographic information. Without the information of the demographic repository, the access to the EHR repository does not leak identification of the patient. The demographic repository acts as a frontend to, or implements, the patient master index. The archetypes are in a repository enabling the use of different versions of an archetype and can make use of terminology services

An OpenEHR consists of a logically organized structure of folders, each containing several versions of healthcare events. Versioning is accomplished through recording every data change in special data structures (contributions). The structure is described in Figure 2.12, followed by the explanation of the different parts [6]:

- EHR – root object, identified by a unique EHR identifier;
- EHR_access (versioned) – contains access control settings;
- EHR_status (versioned) – contains various status and control information about the record;

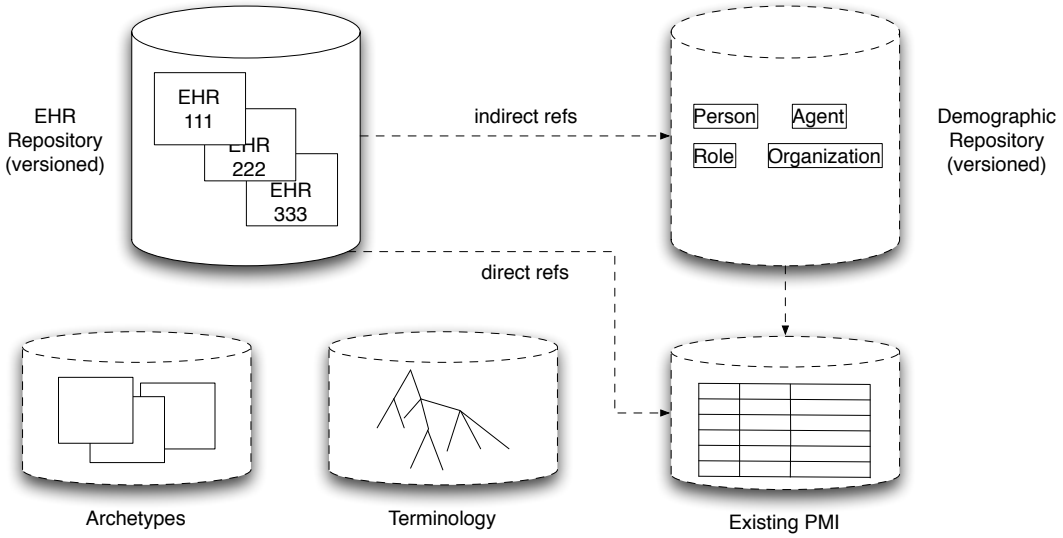

Figure 2.11: Minimal OpenEHR System [6]

- Directory (versioned) – optional structure of folders to logically organized compositions;
- Compositions (versioned) – containers of all clinical and administrative content of the record;
- Contribution (versioned) – contains every change made to the health record, each contribution can refer to one or more versions of any versioned item in the record that was committed or attested in the system.

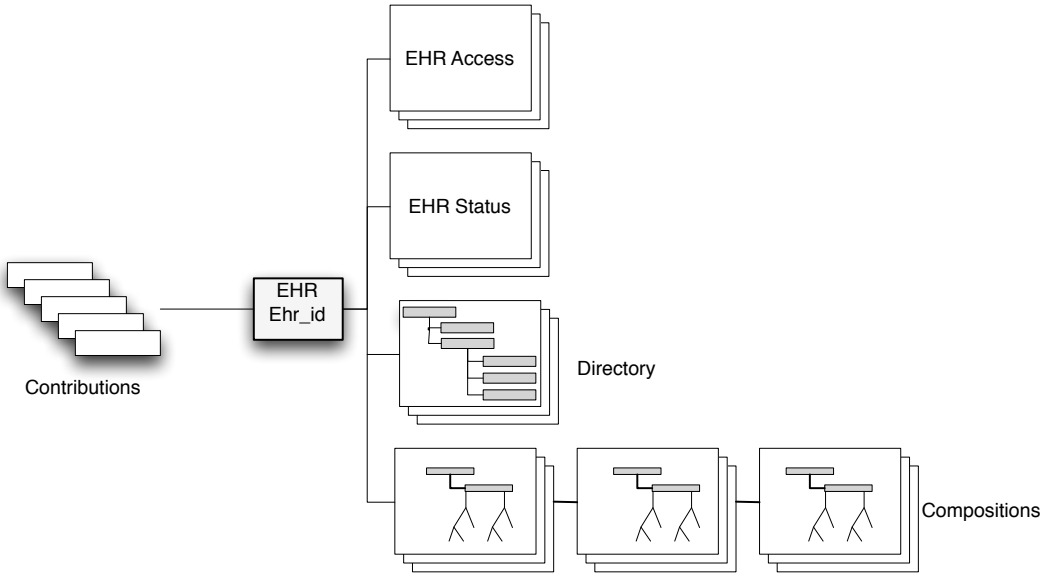

Figure 2.12: OpenEHR structure [6]

The internal composition structure and the directory objects correspond to internationally agreed models of health information, such as CEN EN13606 and HL7 CDA standards [6].

2.2.4 DICOM Structured Reporting

DICOM Structured Reporting (SR) is an extension to the DICOM standard to enable manipulation of clinical reports, adding the ability to generate, distribute and manage reports. It is mainly used for encoding medical reports on a tag based format. A structured report has a header, used also for the DICOM images and the content is represented in a document tree [8]. Parent and child nodes are related through a set of relationships [5] (Figure 2.13).

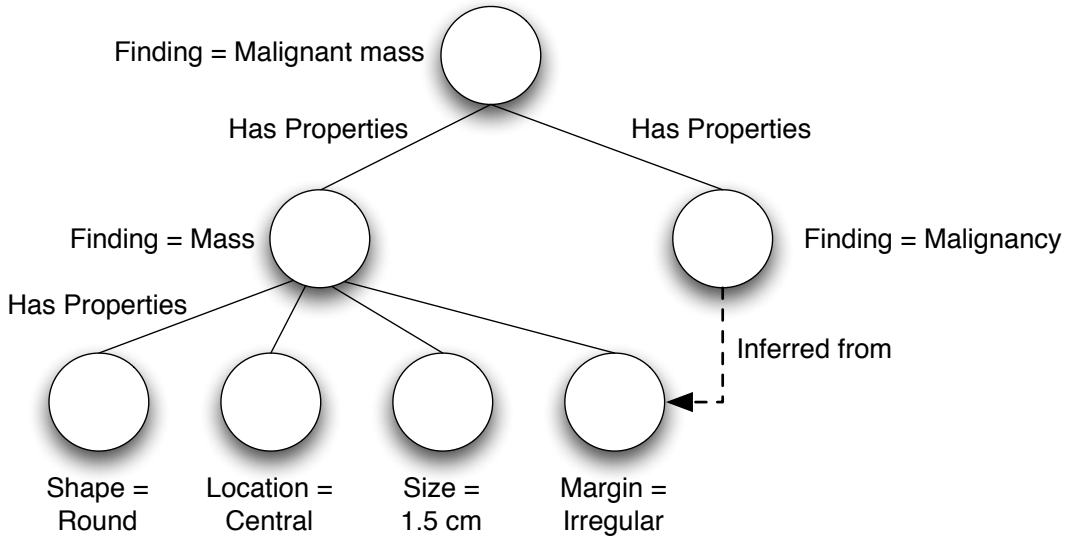

Figure 2.13: DICOM Structure Reporting with References [7]

SR documents can be used to represent structured information, such as lists, hierarchically structured content, numeric values or references to images. In DICOM SR only the meaning of the information is encoded, focusing the main concern on semantics and not presentation. It is limited by not supporting references to other SR documents, since the content must be fully enclosed in a single document. If references to other documents are needed, the information should be copied to the document that needs to refer to it [8].

2.2.5 Integrating the Healthcare Enterprise - IHE

Integrating the Healthcare Enterprise (IHE) aims at facilitating information sharing among healthcare systems, using established standards such as HL7 and DICOM. It tries to overcome interoperability difficulties between standards implemented in different systems. Providers should therefore first decide the need of integration, then use IHE solutions accordingly to enable integration [36]. Two main communication standards are defined: Retrieve Information for Display (RID) and Cross-Enterprise Document Sharing (XDS).

The RID enables the access to persistent objects in standard format as HL7 CDA, Portable Document Format (PDF) or JPEG and other patient information such as allergies, medication or reports across different healthcare institutions [37]. RID defines a web service using a Web Service Definition Language (WSDL), enabling retrieval through a HTTP GET method.

This approach is based on two actors: the Information Source Actor and the Display Actor. The first has the information and the second requires access to the information for displaying it to the human observer. The communication is initiated by the Display Actor by making the

request to the web service. The Display Actor can request to Retrieve Document for Display or Retrieve Specific Information for Display. The first retrieves a persistent document, the second may access information that is updated frequently, such as allergies and medication among others. The phases during the retrieval of a persistent document are:

1. The Display Actor provides an Unique Identifier (UID) to identify the document requested and information related to the formats supported;

2. The Information Source sends the requested document as a payload of the HTTP response. When requesting specific information, the patient ID and the sought type of information is provided by the Display Actor.

The HTTP response is an Extensible Hypertext Markup Language (XHTML) document on the HTTP response payload. The document may contain hyperlinks to other persistent documents which can be retrieved using the Retrieve Document for Display process [8].

The IHE propose the XDS for supporting healthcare documents sharing. XDS is a solution for integrating documents from different producers. The main idea is to store documents in an Electronic Business using Extensible Markup Language (ebXML) format in a Document Repository and registering them in a Document Registry to promote their use [38]. The Document Repository stores the documents and the Document Registry manages metadata to facilitate the discovery process by the consumers. Sharing is possible within a well-defined group of healthcare enterprises which agreed to collaborate, which is designated as an XDS affinity domain.

The actors and transactions are defined on Figure 2.14.

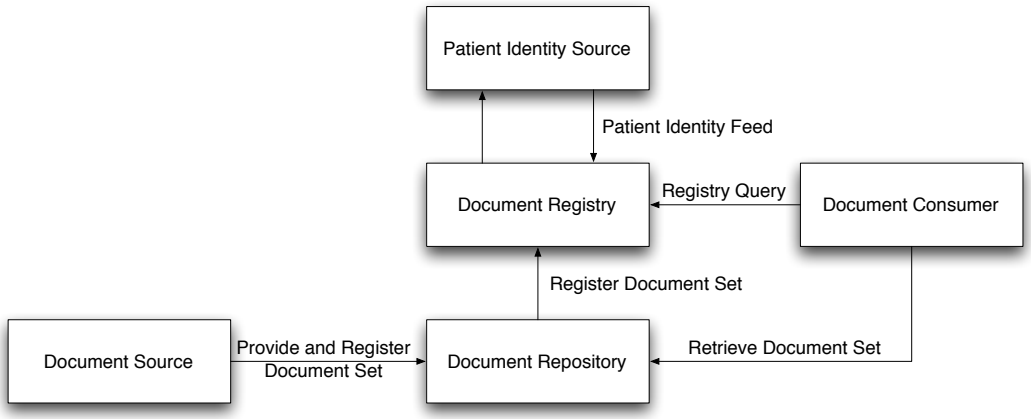

Figure 2.14: IHE XDS Actors and Transactions [8]

The actors are the Document Source, Patient Identity Source, Document Registry, Document Repository and Document Consumer. The Document Source produces the document and sends it to the Document Repository using the method Provide and Register Document Set. Then the Document Repository stores the document and sends the document metadata using the method Register Document Set to the Document Registry.

The Document Registry can answer to queries of the Document Consumer by the method Registry Query for all registered documents. Matched documents can be retrieved by the Document Consumer from the Document Registry using the Retrieve Document Set method.

The Patient Identity Source is used to identify patients to the Document Registry using the Patient Identity Feed.

The document content is not contemplated on the XDS. Documents can exist in any type of format. Therefore, to enable a correct communication, a document format and structure should be agreed inside the XDS affinity domain [39].

2.2.6 Discussion

In this section different standards for communication and document were discussed. Table 2.5 summarizes the differences between communication standards. Almost all of them support querying, retrieving and submitting operations. The exception is WADO which does not support query of document contents and submission of new documents. The IHE RID does not support submission since it is oriented to retrieve the information for display only. The RID and the XDS do not make any consideration of the format of the documents and they are manipulated as black boxes.

Table 2.5: Communication Standards Comparison – Adapted from [8]

Functionalities	WADO	EHRcom	DICOM SR	RID	XDS
Query Service	N	Y	Y	Y	Y
Submission Service	N	Y	Y	N	Y
Retrieve Service	Y	Y	Y	Y	Y
Document Centric	Y	N	Y	Y	Y
Content Format Agnostic	Y	N	N	Y	Y

Document standards consider EHRcom, HL7 CDA, OpenEHR and the DICOM SR (Table 2.6). All of them allow to store structured documents and they also consider the document as the basic unit. EHRcom and OpenEHR, further enable the aggregation of diverse documents in a single composition. Multimedia and reference to multimedia are also supported by all the discussed standards.

All use two level modelling, although with some discrepancies. The EHRcom and OpenEHR use archetypes, while others use templates. Only the DICOM SR has a predefined library of possible objects [8]. The control of distribution rules can only be made in the EHRcom and OpenEHR.

Some standards are analogous, but they also have some ambiguities which further complicate interoperability. The use of some proprietary formats in systems that are already in place makes it even more difficult to achieve interoperability between systems. The DICOM standard is the most stable and consensual in the medical image area. In records area, there is still a solution missing that would enable interoperability between all actors in healthcare. The OpenEHR project has the goal to enable collaboration on the creation and validation of archetypes. This collaboration can foster OpenEHR archetypes to be common to different systems and work as a common format for information sharing.

Table 2.6: Document Standards Comparison – Adapted from [8]

Functionalities	EHRcom	OpenEHR	DICOM SR	HL7 CDA
Store Structured Docs	Y	Y	Y	Y
Document basic unit	Y	Y	Y	Y
Supports Compositions	Y	Y	N	N
Multimedia	Y	Y	Y	Y
Reference to multimedia	Y	Y	Y	Y
Archetypes/Templates	Y	Y	Y	Y
Defined Library	N	N	Y	N
Specify Distribution Rules	Y	Y	N	N

2.3 Integration

Even with the standardization efforts described above, successfully achieving an ICEHR is still a challenge, because information does not reside in one single system. The free market in healthcare where providers compete to provide services bring the possibility of choice to the patient. This aspect combined with the increased mobility lead to the information to be duly disseminated among providers. Citizens change their residence during their lifetime, they travel more regularly, for work, for leisure or even for medical care, hence scattering medical information [40]. However, the provision of quality healthcare services demands a unique view of the disperse EHR.

Solutions are needed to cope with the challenges of scattered data in different institutions, possibly implementing integration mechanisms. One approach is to have an integrator that knows the location of the information and how to retrieve it in a secure way. Some projects decided to extend electronic health card to support that service. Hence, the Virtual Unique Electronic Patient Card (VU-EPR) appeared as a possible solution. Costa et al developed a VU-EPR solution designated Multi-Service Patient Data Card (MS-PDC) [41, 42].

MS-PDCis a smart card that stores, in a secure way, card-owner resident clinical and administrative information. It can also store references in a structured data set to scattered electronic records. The association of Public Key Cryptography and Crypto Smart Cards provides secure methods to store, transport and access the card-owner's information. Moreover, it also grants the owner full control over the access to its data, through a PIN and/or biometric registration.

This MS-PDC model empowers patients and allows discretionary access to remote data. Crossing MS-PDC with health professional card enables professionals to gain access to the medical emergency data stored in the card. The card owner can define information access levels to other users, such as the clinical staff. MS-PDC uses URLs to fetch the information on the disperse systems and present them to the user as a unique view. This model copes well with mobility issues, such as aggregating disperse data and controlling its access.

In a wider concept of mobility, it is not feasible that all worldwide patients will have the same type of card. Another open challenge is to allow services and users to access patients' information when the card is not available. For example, updating URLs cannot be performed if the card is not present. Moreover, losing the card represents loss of all the information it contains.

Some authors suggest that the adoption of web services technology is inevitable [43, 44]. The key factors include the network awareness, integration of legacy components and the path to enforce security policies.

A solution to overcome these disadvantages is to use an analogous but deployed architecture in a Service Oriented Architecture (SOA), making use of a Virtual Health Card Service (VHCS) that mimics the behavior of the MS-PDC [45]. The proposal is supported by a centralized access control mechanism which implements the patients' informed consent through a policy. A proxy makes use of the information in a Virtual Health Card, namely, the credentials of the patient to the repository, the access policy to apply to the requester user, and the URLs to the scattered repositories to retrieve the information and create a unique EHR read-only view [46].

Each repository has to provide query and retrieve operations, in order to gain access to the patients' information. The tight regulatory framework that healthcare providers have to comply with when manipulating clinical data almost makes this requirement impossible. This will also require that all the providers have technical and human resources to support the gateway.

Another approach is to make a central repository available to healthcare providers to deposit the information after patients attend their services, thus paving the way to the ICEHR. The patient may even decide the access control policy to that repository.

The repository can be a PHR, as already discussed. Dossia, Google Health and Microsoft HealthVault enable similar behavior [13] and they provide methods that enable services to upload content to the patient's PHR. As those services are oriented to record management by the patient, they can represent a barrier for the use by medical staff. The integrity of the data is dubious, since the patient can also create, delete and modify data.

Even considering that specific store services will appear, there is the problem of ensuring data privacy, when a patient stores his record in an external provider. This is one of the concerns about PHR privacy, especially when information exists in a controlled environment inside the healthcare provider's networks and systems. Usually providers give access to every EHR, which can cause large-scale disclosures. Even with audit trails, privacy breeches are difficult to be detected automatically, because there is no link between provider and consumer [47]. This dilemma is even more critical for PHR providers which aggregate information for many patients.

An approach to minimize these problems is to cipher data before storing it. One solution oriented for encrypted storage of medical data in a grid using a key to encrypt the data [48]. The key is split and distributed through different key-servers, keeping references to them in the storage service. To get the information, a user retrieves data from storage and the addresses of the key-servers. He queries the key-servers for the sub-keys, if he has the necessary privileges, and computes the key to decipher the information. This approach has the drawback of not allowing to associate different access views to different users. The access control is based on grant access or not. In fact, it is not possible to use a finer grain policy. Another problem is that this approach does enable to maintain a link to the requesters, since it always uses the same key to decipher. This makes it difficult to distinguish between requesters. Besides, as the same key is shared by different users, the possibility of compromising the key increases. Another important aspect deriving from the key sharing, is that an user can retrieve the information from the storage even after the policy has expired, because the key is the same. A possible solution for this is to retrieve the data, delete it on the storage, decipher and undergo a new store procedure from the very beginning.

Another challenge of having a ciphered repository is to provide some indexing capabilities and to allow selective retrieval of data. Some work has been done in searching over ciphered databases, usually through having a separate metadata store [49].

Some countries are making efforts to implement integration at national level. In Portugal, a project named *Registo de Saúde Electrónico*, has the purpose of creating an Electronic Health Record for every Portuguese citizen. It will enable the access to relevant patient information, integrating all the scattered information and enabling the sharing of information, and supporting patient mobility [50].

It proposes using a three level federative model. Each level has different authority levels and also diverse information. Level one is centralized, managed by a central authority. Level two implements a distributed model, which can result of a regional federation or can be managed by the providers. Level three is composed by systems which belong to the providers who support directly operational tasks for the providers.

Information flows from level three to level two, level three to level one and level two to level one. The method is based on the principle that information is produced on level three, and then propagated to lower levels. Access and visualization of information on level one and two should be made, when possible, using interfaces to the systems on level three. If not possible, access portals should be used. When information is pushed to the shared level (level 2) or to the centralized level (level 1), an index in level one is updated to enable future access for retrieving or update.

HL7 and DICOM are currently being considered to be used at least for communication between systems [51]. Level 1 information is also proposed, such as general citizen information, clinical alerts, problems list/diagnoses, among others.

The possibility of patient active collaboration in creating information is being considered in Portugal, through the portal[3]. It gives patients' access to some services and to the creation of information, dessociated from the EHR approach discussed in the previous paragraph. It is still at an early stage, enabling to schedule appointments, request prescriptions. On the contrary, an area patient information creation is not yet available.

Mobility can also cross country borders. In European environment for healthcare the European Patient Smart Open Services (epSOS) appeared. Currently, the services provided are the Patient Summary with general information and medical information, and the ePrescription. Medical information is a summary composed of information related to allergies, implants and major surgeries during the past 6 months. The ePrescription enables a patient to buy his medication in other countries. This service deals with information such as name, code of medication and dosage [52]. The basic architecture requires that each country provides services in their infrastructures – a National Contact Point. Each has a national interface connector (1, 2), specific for connecting to the national infrastructure, as illustrated in figure 2.15.

Health professionals during care delivery can access to the network through the portal or the portal adapter (3). If a country develops its own portal it also needs to implement the portal adapter as a web service. The core component (4) is an IHE X protocol terminator service. The epSOS interface (5) enables the communication between the different member states with other national interface connecters and both have roles for outbound and inbound communication. The architecture is is based on IHE profiles and is implemented as web services [9, 53]. They make use of CEN EN13606 reference models for patient demographic

[3]Available at `https://servicos.min-saude.pt/acesso/`

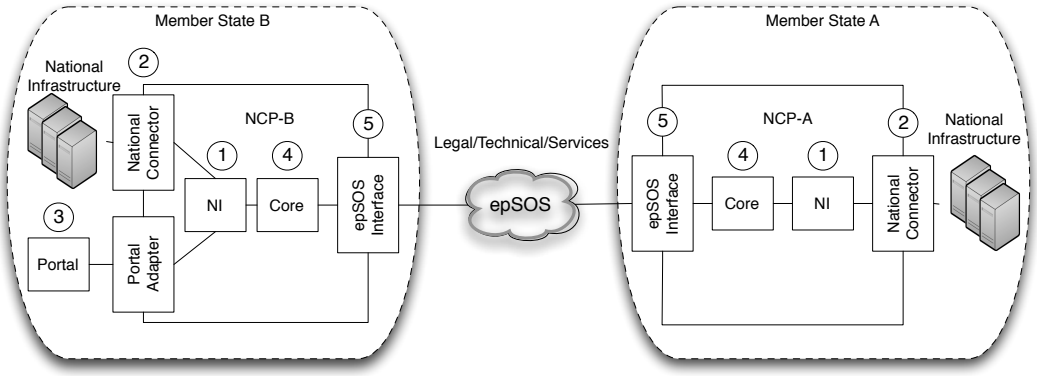

Figure 2.15: epSOS Basic Architecture [9]

and summaries, using archetypes defined in ADL [54].

This approach works when dealing with healthcare professionals who are members of a national network, which belongs to the epSOS network. The collaboration is limited to previous agreements among providers, member states, and states. The patient does not have full control of access to his medical information, he does not have the freedom to give access to everyone he desires and cannot collaborate or validate the information in the repository. Moreover, through this approach, all the states are required to implement a national infrastructure that gives access to the information spreading among all the healthcare providers. Information or actors that are not referred to by the state cannot access the information. The creation of information by all involved actors with the aim of achieving a full integrated record seems unlikely to happen.

2.4 Security Requirements

Information security is of the utmost importance, particularly in healthcare area. Regulation and legislation are needed to ensure data protection, as well as to define the responsibilities and rights of all the actors involved in healthcare data manipulation.

Many laws and codes of ethics are in force to protect the patient personal data, and health professionals. Two competing requirements exist: healthcare professionals need access to patients' information to provide proper care; on the other hand, information should be protected from unauthorized actors.

In Portugal the general law of data protection applies when dealing with electronic information [55]. Access to health related information is allowed when needed for diagnosis and treatment. Access is only allowed if the information is manipulated by a professional bounded to professional secrecy. Other regulations include the Basic Health Law [56], the Doctors Ethical Code [57] and the Nursing Ethical Code. They are mainly good practices manuals to deal with healthcare information by healthcare professionals. They do not describe special concerns for electronic information.

A specific law to regulate the access and manipulation of medical information was approved in January 2005 [58]. This law clarified who is the owner of the information and who has access to it. The law states that the medical information, including medical data, medical tests, interventions and diagnostics are property of the patient. Healthcare service providers

are considered faithful custodians of clinical information and it cannot be used for any other purpose than providing healthcare services and other cases established by law.

Medical information is generated by the doctor who assists the patient or by any other professional under the doctor's supervision, also subject to confidentiality. It is also assumed that the responsible for processing the data should take measures for both data protection and confidentiality. The owner has the right to access all the clinic process related to him, except in special situations – when proved that the information can be prejudicial to him.

Clinical information can only be used by the health system under the conditions expressed, and with written authorization from its owner or representative. When anonymized, health information may become available for research purposes.

The European Commission also made suggestions regarding this subject in 1997, to be found in the Recommendation for Protection of Medical Data to ensure proper safeguard and management to achieve confidentiality, integrity and availability of personal medical data [59]. In 2004, a new recommendation was issued focusing on the use of the information technologies in healthcare, for instance the Internet, and their effects on data manipulation [60]. This also fostered the patient empowerment, enabling him to participate in the collection and access to his own information [61].

Other important regulation effort which is well known and in use at the United States of America is the Health Insurance Portability and Accountability Act (HIPAA), created in 1996 and enforced in 2003. It establishes standards for the privacy and security of healthcare information, as well as standards for electronic data interchange [62].

Architectures which deal with healthcare information should be able to enforce regulations, as well as to enable collaboration between healthcare actors, hence, enabling the creation of a lifelong healthcare record.

The security of a system depends on all of its components and dependencies and having the most secure authentication and authorization schemes is not enough. Communication should be secure and resistant to replay, eavesdropping, traffic analysis, and other network based attacks. Moreover, the systems that support the services (operative systems, network applications, ...) should be tightly configured and all components should be designed, implemented, configured, and used taking security into consideration.

The healthcare area is a rich scenario with various security challenges. From authentication and access control to communication security and cryptosystems [63], security mechanisms should be used to enforce data security. It is a challenge to secure this information especially because of its size, mobility of staff and patients, data sensitivity, and the complex rules which govern the manipulation of clinical data.

The main security principles for a lifelong EHR system is discussed in [64]. Requirements for confidentiality, control, integrity and legal value are briefly stated, considering that patient records are private and confidential. Integrity of the information is crucial, since the life of the patient depends on it: only authorized users can create and update data. For legal purposes the patient records should be tamper proof and should store all actions taken by healthcare professionals

The integrity and availability of the information is very important, therefore, solutions to build more robust and redundant systems should be considered. Records need to be updated as soon as possible. The delay between the creation of new information and its availability should not be significant. Moreover the usability of the record is also important: it should be easy to use, all relevant current conditions, allergies should be easily accessible and search functionalities can improve its use.

Confidentiality and privacy of clinical data have a strong professional tradition. Healthcare professional are regulated by an ethic code that it protects professionals and requires them not to leak patient clinical and related information. Modern healthcare systems brought more actors that need to share and use information. That challenged the confidentiality and privacy of the patient clinical data [65].

The confidentiality of EHRs can be enforced through access control mechanisms and audit trails. The latter are used for inspection of suspicious activities. System users must also be educated about security issues and their responsibilities. Human factors, such as errors, negligence and unethical activities can lead to breaches, despite the implemented security mechanism [66].

Access control mechanisms in healthcare area have to deal with emergency access. Practitioners may have to access the patient EHR when he does not have conditions to grant access. In such situations, systems should enable practitioners to bypass the access control mechanism. This procedure is well known as break-the-glass [67].

Most of the authorization schemes used for regulating access to medical records requires that the patient implicitly accepts the healthcare organization structure. Once the patient authorizes access to a healthcare professional of one organization, the patient accepts further delegation by that professional within the organization. This implies that the patient loses control from the moment a professional inside an organization gains privileges. Another controversial principle is the read access for public health, legal and professional entities. This principle considers that those entities can have a limited, read only access to patient's anonymized data without the approval [64]. In an EHR, those requirements can be less controversial because the guardian of the information is not the patient. But in a PHR perspective all access should be granted by the patient.

As already stated, the legal value of records brings special security concerns [64]. Along with privacy, identification and authorization concerns, it also requires nonrepudiation, so that no one can deny making entries or updates in the patients' records. It should also use an incrementally approach to protect the record from harmful deletion or alteration. This can be achieved through versioning techniques: only the newer version is showed to users, but, if needed, other versions can be recovered for inspection or correction. The storage should last through all of the patient life and even longer, so digital signatures validity is a special concern for future verification of data integrity. Another challenge posed is related to credential's management. Eventually patients will forget passwords and keys, lose smart cards, or other tokens.

Most of the previous requirements are not achieved by most implementations. For example recent developments in PHR do not satisfy integrity and as such legal value [64]. Another problem is the lack of privacy of the information from the store service provider.

To achieve the previous requirements, authentication, authorization, accounting and cryptography mechanisms will play an important role for the proposal of a possible solution. The analysis of security standards for healthcare applications is particularly useful when concerning the uniformization of security approaches [68]. The challenge grows in the context of mobility, as actors from different providers and even from different countries compose the scenario.

2.5 Regulating Access to Health Records

The authentication and authorization processes are fundamental steps to ensure clinical records security. Considering the environment of multiple healthcare providers, possibly from different countries, trust between all actors is also an issue. Joining a new actor to the system should be easy and seamless in order to, also trustfully, enable open collaboration among all the authorized actors.

Authorization is important and implies that the defined policies are correctly enforced by the system. Different approaches are discussed in subsection 2.5.2..

Another important area is accounting to verify access to the records. It will be a core aspect for verifying and maintaining the legal validity of the record. Subsection 2.5.3 is dedicated to this issue.

Cryptography has a great role in enforcing some of the requirements, especially related to privacy and data integrity. Some of the solutions normally used in this area will be discussed in section 2.5.4..

As already clarified and highlighted in section 2.4, health informatics are rich regarding security challenges. The community created special guidelines, such as ISO standards, to address the EHR security challenges. Health Informatics - Privileges management and policy management [69], formal models [70] and the CEN EN 13606-4 Health informatics - Electronic health record communication - Part 4: Security [10]. These standards will be discussed in this section with technical solutions for targeting the security requirements.

2.5.1 Authentication

Authentication is the process of confirming someone's identity. Usually, this is achieved through something he has (card, token, ...), something he knows (PIN, password, ...) or something he is or does (fingerprint, handwritten signature, ...) [71]. Stronger authentication can be achieved by combining different methods, for example, a fingerprint and a password.

Different methods are available for authentication. O'Gorman provides an interesting comparison between them. He considers password, token and biometric based ones [72] and states that passwords are the most widely used methods for user authentication, as it is familiar and simple to use. However, users tend to choose weak passwords, easier to remember and therefore easy to discover through a dictionary attack. Sometimes, users even write the passwords on paper, thus resulting in bigger risks of disclosure. Another problem is the cost of resetting forgotten passwords and the difficulty to realize that a password was compromised. There is no mechanism that triggers an alarm if someone discovers the password of a user. Finally, it has very low defense against non-repudiation.

Tokens can store or generate multiple passwords which can be used in password authentication mechanisms. The main advantage is that the user does not have to memorize them. He may only have to memorize one password, to be able to access the token, if it is protected by one. If the token is missing the user knows that it has been compromised. Besides, the attacker will not be able to guess the passwords: either he has the token or the access will not be granted.

Authentication based on passwords are prone to attacks that aim to guess or discover the combination. This vulnerability usually requires some failure accounting, locking the system after a predefined number of tries. This can lead to denial-of-service, because legitimate users will be out of the system until the problem is solved.

As stated above, stronger authentication methods include two or more forms of authentication. Yet, single-token authentication should not be used to grant access. More commonly, it can be used as a first line of defense protected by a password or PIN or associated to biometric mechanisms. Albeit the latter is more expensive, it saves the user of memorizing a password. The solution will be to use a biometric feature instead.

Biometric authentication has a residual probability of loss or to be eavesdropped, as biometric features are more difficult to lend or to be stolen. However, this option also suffers from the possibility of false positives: recognizing the same biometric features from different subjects.

Usually, biometric features have two types of signals: stable and alterable. The stable signals are more prone to be stolen and copied and alterable. Hence, they are more resistant to forgery and replay, but they have the inconvenient of having more false non-matches (avoiding to recognize the correct user).

For the above reasons, biometric authentication methods should not be used in a single-factor mode. They should be combined with other methods and only used as a second-factor authentication, for example with smart cards and Public-Key Infrastructure (PKI). Moreover, a person's biometric features are stable and distinct so, if it gets compromised, it is harder to change.

Biometric authentication can be based on a large set of different features [73] such as face, Facial Thermogram, Fingerprints, Hand geometry, Retinal Pattern, Signature, Iris, Speech and many others. Each one of those techniques has a specific level of usability, error incident, accuracy, cost, user acceptance, required security level and long-term stability. One of the biggest problems associated with biometric authentication is surely the way to store the features. A possible solution is the use of smart/crypto cards. Hence, recent type of cards can secure store information using a cryptography built-in fingerprint reader to generate the key used for cyphering and deciphering the data.

Another method of authentication makes use of a PKI. Hence, it uses public-key cryptography for identification and authentication. This method is mathematically more secure than biometrics and copes with Internet usage. However, it has the problem of private key management, as it should be portable and protectable. Therefore, solutions to enhance security and privacy in biometrics-based authentication systems [74] are being studied. Some solutions store the private key on a smart card protected by a biometric feature.

Among the previous discussed authentication methods, smart cards have been proposed for several scenarios in the healthcare area. For instance, they are considered good solutions for authentication [75, 76, 77] and for storing information to enable remote access to systems [75]. Hence, the card can be used for multiple purposes, integrating the authentication and information that need to be portable and with the user. Moreover, some also integrate a certificate which can be used for the authentication process. In addition, some authors contemplate the use of government issued digital identification cards as a solution that could be used in Health Information Systems [78].

Carlos referred that a smart card could be used as the token for identifying health professionals and patients [75]. In this approach, users could use a PIN or fingerprint to validate them as card owners. Further, the card stores patient information and URLS for web services where the patient's clinical information exists. In brief, the card is a repository of links to disperse healthcare information stored on the healthcare centers. However, it has some drawbacks related with the use of tokens, such as problems to access patient information if the patient forgets his card. Besides, the loss of the card or the need of its presence on the system

for some procedures as discussed in section 2.3. Finally, the cost of creating and maintenance of a PKI is huge and difficult to implement in a worldwide scenario.

One important aspect of the use of smart cards and certificates for authentication is the trust on the certificate. Therefore, PKIs have great importance, since they are responsible for validating the information and identity of certificate requesters. Thus, if the burden of the PKI management could be carried by a reputable actor, the trust on the certificate would increase. Further, several countries are moving to Electronic Identification (eID) cards. Those cards have certificates signed by the country PKI that enables strong identification and authentication of a citizen. Thus, eID solutions enables citizens to make self enrollment in systems with high trust of the identity of the user, even without the system being aware of information about the user, as long as the system trusts the users' country PKI. In Portugal, the Citizen Identity Card has strong authentication and digital signature capabilities. It is a secure card, with latest encryption and protection technologies. Moreover, it also supports a RSA PKI managed by Portugal, enabling remote authentication, thus making it good for use in Health Information Systems [79].

The Portuguese Citizen Card and other countries similar electronic identification cards became good choices to authenticate remote users and even for self-enrollment in systems.

2.5.2 Authorization

After authentication, systems have to build an authorization profile, in order to describe the capabilities and resources associated with a given user.

The most common authorization mechanisms are based on access control lists, such as Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role Based Access Control (RBAC) [80, 81, 82]. Access control lists are built on the concepts of subject and object: the former are who/what accesses (a user or a process) and the latter is the accessed resource (files, pipes, network devices and others).

In DAC the access policy is determined by the owner of the object. The user that creates the object becomes the initial owner and is responsible for defining the corresponding access policy. In other words, the owner will decide who is allowed to access the object and what privileges they have.

MAC is based on access policies determined by the system and can handle different classification levels between subjects and objects. It is usually used in multilevel systems that process confidential data, such as classified government and military information. It depends on the principle that every object and every subject has assigned labels (sensitivity labels) to. The subject label specifies the level of trust. In order to access an object, the subject has to have a label with the same level, or higher than the object. This access policy is used for controlling the import and export of information from or to other systems. The most commonly used methods for implementing MAC are the RBAC and the Lattice-Based Access Control [83]. The first determines if access should be granted or denied, by matching the sensitivity label of the object with the subject label. The second is used for complex access control decisions, defining a lower-bound and upper-bound value for each pair of elements, such as a subject and an object.

The access policies are defined by the systems in MAC as in RBAC. The system defines the sets of permissions that may include operations, e.g. e-commerce transactions or a simple read and write. Moreover, this access policy has its pillars on the following three primary rules:

1. role assignment: a subject can execute the transaction only after selecting or assigning a role;

2. role authorization: the active role of the subject must be authorized. Together with the previous rule, this ensures that users can only be assigned to roles that they are authorized to;

3. transaction authorization: a subject can only execute a transaction if authorized for the user active role. Along with the previous rules, this insures that users can execute only transactions that they are authorized to.

RBAC is more powerful than MAC, in the sense that more constrains can be applied. More, roles can be combined in a hierarchy where higher-level roles include permissions from the sub-roles.

A variation of the RBAC is the Rule-Set Based Access Control (RSBAC) [84, 85]. This framework can implement almost any access control scheme and has more flexibility than the RSBAC, whose functionality is built as modules, implementing different models or aggregating groups of functionalities. Modules can be combined to achieve diverse access control models, supporting MAC, ACL, User Management or custom made models (Table 2.7) [86].

Table 2.7: RSBAC Modules

Module Name	Code name	Short description
Authenticated User	AUTH	Authenticate Users
Role Compatibility	RC	Role based access control
Access Control Lists	ACL	Extensive Access Control Lists
Mandatory Access Control	MAC	Multi Layer Access Control
Pageexec	PAX	Prevention against unwanted code execution
Dazuko	DAZ	On-access anti-virus scanner
Linux Capacities	CAP	Manages Linux Capacities
Jail	JAIL	Encapsulation of individual processes
Linux Resources	RES	Manages Linux Resources
File Flags	FF	Set special access control flags per filedir
Privacy Model	PM	Controls data privacy in conformance to EU laws

The previous present authorizations methods need to be chosen wisely, because each one have features and behaviors that are tailored to different proposes. Therefore, uses of those methods in the healthcare, especially to control access to electronic health records are further studied. One example of a project that targeted to use an authorization model oriented to coloring access to electronic health records was the PING project. It defined a file system structure for saving the health information in XML files, under role based access control [87]. Each file has a header, describing the access permissions, enabling the user total control to his medical file. Using XML for saving the file allows an easy method for accessing the information via a web server. Also, file compatibility problems are minimized because the information is saved in a plain text file with XML tags. In this approach the privileges included create, read, modify, delete or annotate. The role can be a specific person, an identity which can change along the time (my psychiatrist, for example), or a group of individuals (the physicians of

an emergency department). There is the concept of the data owner (the patient), author (practitioner who created the information) and others.

The choice of using RBAC as authentication mechanism for controlling access to electronic health records is common [88, 89, 90] and the privileges of each role to each object are stored in a structure. Then when a user requests access to an object, he will need to have successfully gain access to a role that has privileges. This behavior enables that person to have templates of roles to objects and then access control administrators can just choose what roles each user can gain access. The use of RSBAC is widely diffused for controlling access to EHR. In fact, the standard EN 13606-4[10] makes use of it, defining, first the types of information sensitivity for record components (see Table 2.8).

Table 2.8: Values of Sensitivity for each Record Component [10]

Data Sensitivity value	Sensitivity level	Description of intended access
Personal care	5	Patient himself and one or two other people trusted by him
Privileged care	4	A very small group caring intimately care to the patient
Clinical care	3	Default for normal clinical care access
Clinical management	2	Wider range of personnel not all of whom are actively caring for the patient
Care management	1	Wide range of administrative staff

These sensitivity levels can be seen as labels that can be associated to each object that the infrastructure deals with. This approach enables the creation of consistent policies, since it has the policy specified for a certain type of object, grouped by their level of sensitivity.

Another important aspect in RBAC is the role that the access control mechanism needs to deal with. In the EHR context, the standard considers the roles explained in Table 2.9.

Table 2.9: List of Functional Roles [10]

Functional Role	Brief Description
Subject of care	Principal data subject of the EHR
Subject of care agent	e.g., parent, guardian, carer or other legal representative
Personal healthcare professional	Healthcare professionals that are closest to the patient
Privileged healthcare professional	Nominated by the subject of care OR nominated by the healthcare facility
Healthcare professional	Party involved in providing direct care to the patient
Health-related professional	Party indirectly involved in patient care, teaching, research
Administrator	Any other parties supporting service provision to the patient

The roles are associated with the functions that each user performs related with the health-

care related activities. This information enables to simplify the creation of the access control mechanism. As it specifies the levels of sensitivity, which will aggregate the objects that have that kind of information, together with roles which will represent all types of possible authorized users. Therefore the structure that maps the privileges that each role has can be specified in a simple table. Table 2.10 shows the allocation of the privileges of each role to the sensitivity level of information.

Table 2.10: Mapping Functional Roles to Sensitivity of Record Components [10]

Functional role	Record component sensitivity				
	Care Management	Clinical Management	Clinic care	Privileged care	Personal care
Subject of care	Y	Y	Y	Y	Y
Subject of care agent	Y	Y	Y	Y	Y
Personal healthcare professional	Y	Y	Y	Y	Y
Privileged healthcare professional	Y	Y	Y	Y+	Y++
Healthcare professional	Y	Y	Y		
Health-related professional	Y	Y			
Administrator	Y				

NOTE 1 – Y indicates access will be granted unless dictated by other policy constraints
NOTE 2 - + indicates access will be granted if the EHR requester is a member of the same specialty or clinical service in which the record component was created. This access may also be granted in healthcare emergency situations if so authorized.
NOTE 3 – Y++ indicates that access to Personal care information may sometimes be granted by mandate to Privileged healthcare professionals in some care settings.

The present approach allows the simplification of access control development, configuration and administration, mostly because programmatically the systems will deal with predefined roles and sensitivity levels. All the business logic of the systems can make use of this simplification. The custom approach is when creating new data to associate it to a level of sensitivity. Each user can request access to a role, based on the function that he is providing. Then considering the roles that the requester has and using the mapping structure the access to the object is either granted or not.

Although different approaches exist to implement access control to health records, some problems still persist. Some authors state that access policies are locally defined and not enforced in the health network. Others, mentioned problems related to the use of obsolete technologies, the lack of system integration, negligence with security information (passwords, tokens, and so on) and forgetting to close sessions [91], thus resulting in difficulties to enforce the access control policy.

A simple solution for access control, orientated for a patient centric controlled policy needs to be proposed. It should address the requirements of openness, mobility and secure access and yet be coherent with ISO standards for the EHR.

2.5.3 Accounting

The use of strong authentication methods combined with a proper access control mechanism can improve the security of a system. However, monitoring mechanisms should be in place, to ensure that processes are behaving as expected.

Accounting can be seen as a registry that stores information of who and how a system or service was used. This detailed log can be used for charging resources' usage, for verifying the correctness and execution of a service, a system, or a user behavior, as well.

Systems that manipulate critical information, such as patients' healthcare records, should provide a robust and trustful accounting mechanism such as clearly identifying the requester, the time of the occurrence, recording requests, detailing every operation and associating it to data operations, e.g., creation, access or update. Some healthcare systems allow bypassing the authorization process, in situations of extreme urgency. When this procedure occurs, systems should have a log with auditing information of the access. This log will be critical to decide if the access was abusive or, otherwise, justified.

The Internet Engineering Task Force (IETF) has been debating accounting and its use in auditing. Auditing can be defined as *"the act of verifying the correctness of a procedure. To be able to conduct an audit it is necessary to be able to definitively determine what procedures were actually carried out to be able to compare this to the recommended process. Accomplishing this may require security services such as authentication and integrity protection"* [92].

The aspects related to cost allocation and billing can be useful if the users should pay a fee related to the resources usage. The auditing process is very important to security, as it enables to verify the correct use of systems, as well to verify that the system is coping with the access policies.

Authentication, authorization, and accounting is usually performed with the help of specific protocols such as the RADIUS, Diameter, TACACS and TACACS+. These are more oriented for network and system environments and are not well designed for application layer.

Accounting models oriented for SOA have been discussed previously [93]. In this environment, service composition is a possibility in which a business process may depend on more than a single service call. In this scenario all the events have to be combined and related. If a user makes a request to a service, and this service needs to invoke others, all the events requested on all services should be recorded together. The auditing facility of the accounting is important to check correctness of the operation of complex systems and specially to verify access to sensible information.

The auditing information can be used for verifying the correct behavior of users, systems and services. Considering systems that have critical information a continuous auditing can bring advantages to enforce the correct operation. Continuous audit is *"a methodology that enables independent auditors to provide written assurance on a subject matter using a series of auditors' reports issued simultaneously with, or a short period of time after, the occurrence of events underlying on the subject matter"* [94]. Some work has been done to enable continuous audit based on SOA [95] architectures. Such approaches can enable a continuous monitoring of every service. Furthermore, it will allow checking if collaboration is respecting the initially specified policies and if some anomalous activity occurs services can receive feedback from

the auditor to react.

Given the importance of securing the services provided to patients, it is of the utmost importance to maintain a robust and tight accounting mechanism. Integrity of logs should be maintained, with a clear identification of the user and the resources he used. Log privacy is also an issue, since it records information that is potentially sensible. Moreover, the possibility of service composition in SOA scenarios requires an integrating and secure accounting mechanism.

2.5.4 Cryptography

As mentioned previously, privacy is mandatory in a secure architecture. It is ensured through cryptography, a means to conceal the message from unauthorized parties. Cryptography goes back many centuries. This term originated from the greek terms *crypto*, that means hidden or secret, and *graphy*, that means writing. According to the etymology, it means hidden writing or secret writing [96]. It can be seen as the usage and the study of methods for achieving a secure communication in the presence of adversaries in a non-trusted channel. Different protocols were created to achieve information security, namely confidentially, integrity and authentication [97].

Cryptography includes the encryption and decryption of information (Figure 2.16). Making use of an algorithm and a secret key, information can be cyphered, therefore, enabling its privacy which will be transmitted using an insecure channel. On the other end, using the reverse algorithm and the secret key, the information can be deciphered. The security of the data depends strongly on the secrecy of the key as well as on the algorithm. This type of encryption is called symmetric key, since it uses the same key for both ends of the communication channel. One of the main aspects is that the key has to be shared among actors in a secure way.

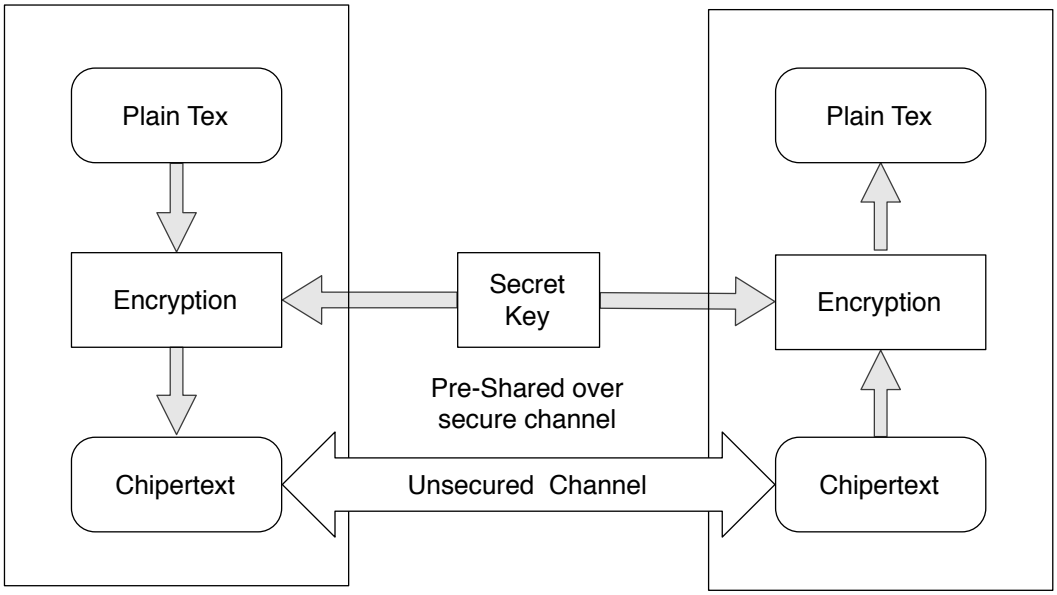

Figure 2.16: Symmetric Cryptography

Usually, cryptography algorithms are block based or stream based. The block cypher

works using a fixed length group of bits. A common block has 128 bits which are converted into 128 bits cyphered information by the algorithm. One of the most used algorithms is the Advanced Encryption Standard (AES). It supports 128 bits blocks with keys of 128, 192 and 256 bits length. It was based on the Rijndael algorithm [98]. Other well known algorithms, although less used nowadays, include the Data Encryption Standard (DES) and triple-DES with single, double and triple keys [99].

Stream based cryptography is generally applied to bits instead of blocks. Algorithms can be designed to be extremely fast, and produce cyphered text as the result of a combination of a variable bit key with the plain text, usually using a XOR operation [100]. It requires a key stream, independent of the plain text, before being used. Typically, a big, one time pad, key is generated, with the same size of the plain text, and is normally used once. The use of such solutions has perfect secrecy, but it also suffers from the problem of key distributing. Solutions like code-books, shared by both parties through personal distribution methods, were used during wartime. Others used extremely secure diplomatic channels. The distribution of the keys makes their use unpractical [100]. One of the most used algorithms is RC4.

Cryptography is not only used for privacy. Another important application of cryptography algorithms is in hash functions and Message Authentication Codes (MAC). Hash function are one way functions that return a hash value or bit string calculated over a block of data. An ideal hash function should not be able to restore the block data from the calculated hash value, enabling the creation of a unique value for the data that can be used to check data integrity. That unique value is also called a digital fingerprint. If data and fingerprint are sent to a receiver, he can calculate the hash of the data received and compare to the fingerprint: if it is equal the data was not corrupted.

Some algorithms are susceptible to attacks, such as the MD5 and SHA-1 that have been proven to have collisions, allowing different data to have the same hash [101, 102]. This is, in fact, vulnerable and these algorithms should not be used for signature purposes. The National Institute of Standards and Technology has proposed alternatives such as the SHA-256, SHA-384 SHA-512 [103].

MAC are typically used to authenticate messages, by creating a hash using a private key and the message data. It generates a hash (the MAC) that is sent to the receiver together with the message. The receiver authenticates the message by replicating the process.

Symmetric cryptography, although using computationally fast algorithms, has two main problems. One is the key distribution problem and the other is validating the origin and the validity of digital documents.

Asymmetric cryptography, also known as public-key cryptography emanates as an answer to those challenges [104]. It involves two separated keys, in contrast with only one in the symmetric cryptography, with consequences related to confidentiality, key distribution, and authentication. A public-key encryption scheme has six ingredients [96]:

- Plaintext: readable message or data to be fed in the algorithm;
- Encryption Algorithm: that will perform transformations on the plaintext;
- Public and Private Keys: pair of selected keys – one is used for encryption the other will be used for decryption;
- Cyphertext: scrambled output of the algorithm. It depends on the plaintext and the key;

- Decryption Algorithm: uses the cyphertext and the matching key to produce the original plaintext.

To start using asymmetric cryptography the user has to generate a pair of keys (public and private keys). Then, he has to make the public key available to the other users, usually by registering it in a public registry. The private key has to be protected and secure under the user's control.

For encryption, user A can send a private message to B by encrypting it with the public-key of B. Then user B, and only him, can decrypt the message using his private-key.

For sending an authenticated message from A to B, the former generates a digest of the messages and uses his private key to encrypt it. User B use A's public-key to decipher the digest and compare it with the independently computed digest for the message received. If digests are the same, he is sure that the message came from A, since only he has his private-key.

The two mechanisms described above are combined for confidentiality and message authentication(Figure 2.17). A cyphers using his private-key a piece or the complete message and then cyphers using B's public-key. B deciphers using his private-key (confidentially, only he can do it), and then deciphers using A's public-key (message authentication, only A could cyphered the message).

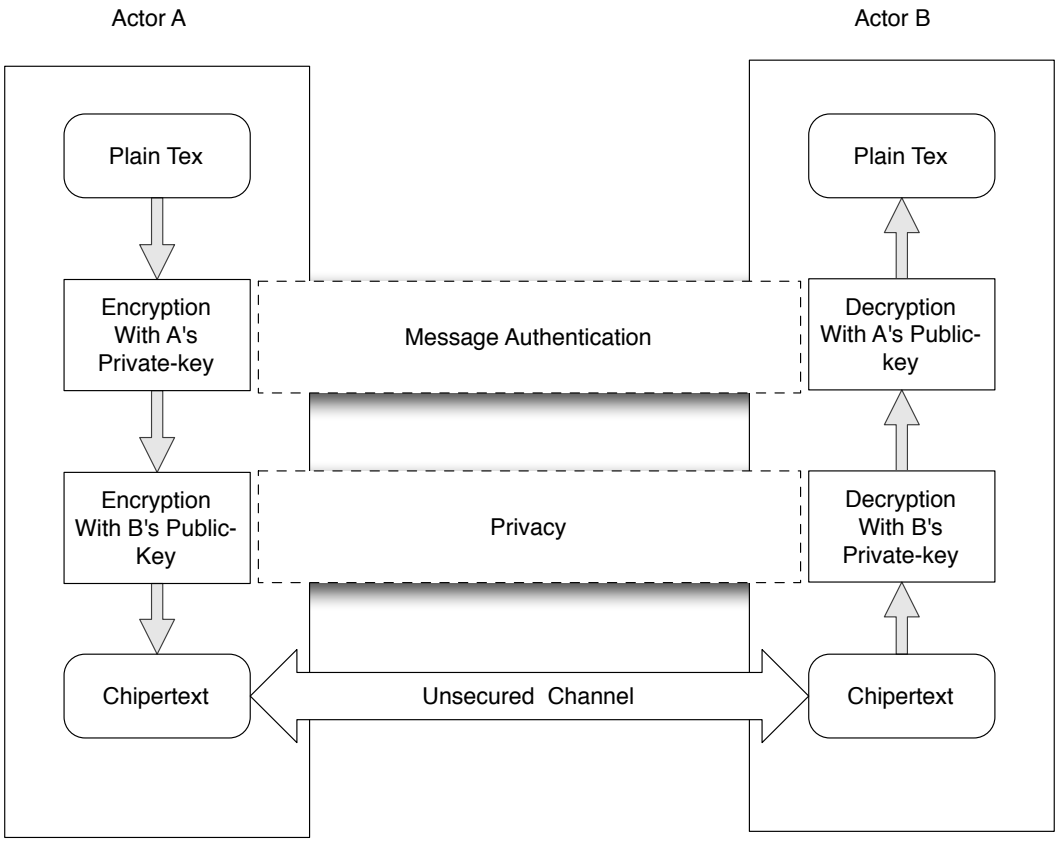

Figure 2.17: Public-Key Cryptography - Privacy and message authentication

The public-key crypto systems can be classified in three categories [96]:

- Encryption/Decryption: sender uses recipient's public-key;
- Digital Signature: sender signs a message with is private-key. It can apply to all data or to a portion (signing the result of an hash function over the message);
- Key exchange: two actors need to exchange a session key.

RSA and Diffie-Hellman are two of the most popular asymmetric cryptography algorithms. The first was developed in 1977 by Ron Rivest, Adi Shamir and Len Adleman at MIT and first published in 1978 [105]. Diffie-Hellman supports key exchange and the RSA supports encryption/decryption, digital signature, and key exchange.

Public-key crypto systems have to respect some computational conditions. It should be easy to compute the public/private keys pair; it should be easy to encrypt/decrypt using the correct key; and the key directive for the algorithm is that it should be computationally infeasible knowing the public-key to determine the correspondent private-key.

The use of public-key cryptography enables anyone to create his key pair and share the public-key. However, making a public widely available does not guaranty that it really belongs to the intended user – trust issues. The solution to the problem is public-key certificates.

A public key certificate consists of a public-key with a user ID of the key owner, and all this block signed by a trusted third party. Normally, a trusted third party is a certificate authority (CA), trusted by user community, a government agency, financial institutions or other competent authority. The CA receives the public key of the actor in a secure way, validates the information contained in it and then signs it. When other users receive a certificate they can check if the information is correct by checking the signature of the certificates. The X509 standard defines the structure of these certificates, enabling the use of them on almost all network security applications (IP security, secure socket layer, secure electronic transactions, and S/MIME). They can be used for signing and cyphering, as well as authenticating users [96].

Due to its role, CA is a component of a PKI. The PKI is responsible for all procedures involving public key cryptography and enabling the use of digital signing and authentication schemes. It should provide services for public-key binding, i.e. validate and signing the actors' public-keys [106]. It should support the revocation of certificates and procures to verify revocation status of the certificates. The PKI has a revocation list where every revoked certificate is announced, enabling services to automatically check the validity of a certificate. The PKI is very expensive to maintain and the key point is the trust that the actors need to have on the CA.

Countries are starting to issue digital identification cards that are based on a country PKI. Identification cards have signed certificates by the country's CA, inside a smart-card or a crypto card. The countries maintain and support the cost of a PKI.

Digital signature schemes make use of certificates. These have expiration date, when the time of storage and use if higher than the expiration date, the signature will be marked as invalid. Thus, long time archiving for digital signing documents need to deal with this type of challenges.

2.6 Summary

The bibliography analysis reveals good efforts to achieve an EHR that enables storing clinical information generated by healthcare professionals within a healthcare network. Sev-

eral standards where proposed and are in place. Moreover, some create proprietary solutions that augmented the heterogeneity of the systems and record formats that manipulate citizen's clinical data.

One of the big challenges that EHR systems and formats face is the constant evolution of the requirements combined with the increased mobility of citizens and with health-care providing liberalization that brought more players and procedures to citizens. Consequently healthcare data was lead to become scattered among the diverse providers. The efforts of integration and interoperability are being addressed by standards, but deployed in roughly enclosed health care scenarios. Strict regulatory, ethic and legal issues have been hindering the wide adoption of EHRs, contributing to delaying the establishment of a competitive market where different providers could take full advantage of information exchange and regular practitioners' collaboration. Therefore, the creation of a real longitudinal lifelong healthcare record demands the open collaboration between all healthcare providers and the patient himself. A collaborative approach seems to be the sustained way for achieving a real longitudinal lifelong health record. Moreover, with the increasing awareness of medical subjects, patients are demanding more control over their own personal data and active collaboration.

To cope with these new user requirements several Personal Health Record (PHR) solutions have been developed which allow users to keep record of their own medical data. Examples of such system are, for instance, Google Health [13], Microsoft HealthVault [13] and Dossia [13]. These web-based PHRs are mostly based on a central repository and on a set of core features that, sometimes, can be extended by external third-party services.

The exchange and storage of health information are a major security challenge because its disruption may compromise seriously personal privacy. Albeit legal constraints associated with this type of data have been largely regulated [14], which imposes restricted rules on the development of technical solutions. The *Big Brother* scenario emanates whenever centralization is suggested, regardless of the guardian of the information being an enterprise or the government, and this vision also slows down the adoption of any different solutions.

In the traditional approaches the patient cannot exercise full control of his record since the providers are not only limited by patient authorization, but also depends of agreements between the repository provider and the healthcare providers or other previous agreement to enter on the network or federation. Also, information clinically related, but not strictly clinical, is not normally stored on the record. The record can be improved with more information that in the future can become important for new procedures or correlation.

To be effective the collaboration needs to open, secure, and support future evolutions. In the following chapter a solution is proposed to achieve a secure solution for an open collaboration between all the actors. The collaboration is only limited by the policy defined by each citizen; the specification uses open standards to provide a manufacture independent solution, giving the patient the freedom to choose healthcare and repository providers and to the providers to develop their services to manipulate patient information.

Chapter 3

Enabling Secure Collaboration

In the background section we have raised several challenges in several directions, but the big question is still how can we build and deploy electronic health records systems that can cope with mobility? First, the dispersion of data among all actors is a challenge. Second, the increasing supply of healthcare providers and new types of providers and procedures makes it more complicated to have systems enabling access to information when needed. In fact, traditional systems and solutions cannot cope with the requirements of nowadays healthcare provision. Considering the analysis made, we have identified that mobility is a specific problem caused by the lack of an open and secure way of collaboration between all actors, including the patient. Therefore, this work proposes a solution to enable this collaboration.

This chapter lists the requirements of a secure solution for open collaboration between all the actors related to the provision of healthcare services, limited only by the policy defined by the citizen. Using open standards to provide a manufacturer-independent solution, we advocate that this will enable the creation and use of a true lifelong electronic health record.

The solution should provide a way to allow information produced by clinical staff, patients and others to coexist in the same record, without questioning the integrity of the information. This solution is based on combining electronic health records and personal health records.

Electronic health records provide a trustworthy clinical record with high integrity of information. The personal health record brings patient empowerment, controlling access to his repository, allowing the patient to annotate and create his/her own information. It also enables non-clinical actors to complement and produce information that can be used in future care, such as sports monitoring, remote monitoring, patient daily log and others (Section 3.1).

To allow truly open collaboration, the solution should employ the most common technology used in industry to promote the development of new services. On the other hand, those services can also contribute by providing new data format and interoperability methods which can foster innovative uses for the available information. To achieve this goal, open standards should be used whenever possible.

Access control requirements are related to any provider's ability to access patient records, when properly authorized by the patient. Thus, one problem is how to balance the requirements of openness and trust. As seen in the background analysis, digital identification cards could be used for secure authentication.

With such an open and collaborative environment the solution should also enable services for monitoring the correct use of all components. Services should enable accounting especially with auditing facilities. Enabling the verification of correct enforcement by the systems of

the access policies and correct execution of all system components.

The patient should be free to manage his access policies. Managing privileges in such a dynamic context can sometimes be a difficult task. Policies need to be effective with the possibility of applying them in a way that patients understand what they are deciding. The aim of this proposal is to create an architecture that copes well with security concerns related to data integrity and privacy, but which, at the same time, should be based on open standards to enable open collaboration. The architecture is presented in Section 3.2.

3.1 Hybrid Electronic Health Records

This section introduces the concepts behind the hybrid Electronic Health Record (hEHR) and how it can contribute to solve some of the challenges, namely the ability of collaboration from all actors.

The hEHR tries to combine features of the EHR and PHR. It can be described as a collaborative record, a service on behalf of the user. Data can be created by the patient and all actors that interact with him, and the access policy is defined and controlled by the patient.

The hEHR is based on a centralized repository, trusted by the patient, to store his contributions. The collaboration is illustrated in Figure 3.1, where the different actors contributing to the creation of the patient longitudinal patient-centric EHR can be seen. Actors who can contribute to and make use of a patient record are diverse and dispersed in a worldwide scenario of mobility. They include the patient, clinical professionals in a healthcare institution and private practice. Also research services, alternative medicine actors, complementary services and others can benefit from the collaboration. Hence, they contribute to achieving a richer and more complete lifelong healthcare record.

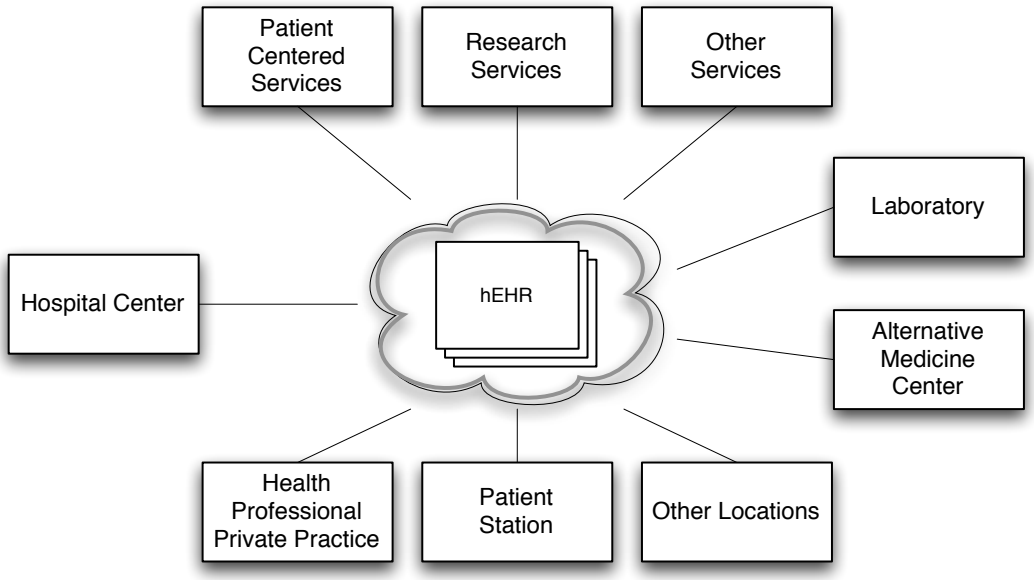

Figure 3.1: hEHR actors'

The patient can control other actors' access through his station, where he can also make

contributions to his record. The healthcare provider in a healthcare facility such as hospitals, laboratories and other medical centers can contribute by exporting reports from their systems or by using external services that create the reports. All actors can make use of the contribution when previously authorized by the patient. New services can manipulate the information as patient centered services, e.g.:

- prescription alarms or other treatment alarms;
- research services which can be used by researchers when authorized by the patient;
- other new services which can bring added value to the use of patients' clinical information.

The patient will be empowered, since he can control who can access and how to access his medical information. Simultaneously, he can easily give actors in the healthcare or similar area access to his information, thus enabling the deployment of new services that can make use of that information. Each new producer or consumer that wants to gain access to the patient EHR needs only patient authorization. The clinical integrity of contributions can be confirmed, increasing healthcare professionals' trust in the system.

3.1.1 Document Format

As debated in Section 2.2, some available document standards include EHRcom, HL7 CDA, OpenEHR and the DICOM Structured Report. All of them enable the storage of structured documents and the document is the basic unit, but the EHRcom and OpenEHR also enable the aggregation of diverse documents in one composition. Multimedia and reference to multimedia is supported by all the discussed standards. Besides, all make use of two level modeling, with some differences; the EHRcom and OpenEHR use archetypes while the others use templates. This is the difference of how each one enforces constrains. Constrains on the generic model in HL7 is made generating more constrained Redefined Message Information Model and Common Message Element Types with further constrains from HL7 templates. On OpenEHR constrains are enforced in archetypes, and a set of archetypes for a specific data collection can be aggregated using templates [107]. Only the DICOM has a pre-defined library of possible objects. Control of distribution rules can be made in the EHRcom and OpenEHR, while the others lack this functionality [8].

Another important aspect is that many of the formats support serialization to an XML document. The use of XML enables data to be manipulated with common tools and allows document structure to be stored in the same file as the data. The type of format of the records is transparent, and producers and consumers have to agree on a common format to be able to collaborate in creating the full EHR record. If not, they must support export/import to/from one common format, enabling a common understandable format to exist in the repository.

In our approach, actors' contributions will be stored as serialized in XML documents. The structure of the record can give information about the type of contribution existing in the record. Even if the data is ciphered, access to the structure gives information about the saved information. Especially, in cases that use a specific template for saving a result of a lab test or medical procedure can reveal that the patient made that test or procedure. Hence, with some mining over the structure of the full record, guesses about pathologies and chronic diseases can be made. In order to mitigate these risks, the XML documents will be cyphered

to provide data and record structure privacy, ensuring that the structure does not disclose information about the kind of information existing in a patient record.

The intention of OpenEHR project to enable collaboration in the creation and validation of archetypes could allow OpenEHR archetypes to be common in different systems and work as a common format to share information. Also, new archetypes for OpenEHR can be created as new services or care providers can manipulate their information in an OpenEHR repository.

The OpenEHR can deal automatically with this serialization, and then common XML APIs can deal with documents which will be in transit. Therefore, the manipulation of this information will be accessible to a wider group of solution providers. Use of the OpenEHR will also enable new requirements for any group of providers to be answered, as they can develop new archetypes in collaboration and share them with all participants. This is already done in clinical archetypes. A repository exists which all users can contribute to and use. Thus, this philosophy can be extrapolated to new groups of providers with different requirements.

The OpenEHR format will also enable classification of the information directly on the document itself, since it has a header for dealing with security constraints and information classification. This classification will require the proposal of an expedite solution for access management by the patient.

For those reasons and also the openness of the OpenEHR project itself, the proposed solution contemplates the OpenEHR as a document format for all actors. Almost all components of the architecture will deal with a bulk of XML data, enabling other formats to be deployed with few changes.

The future of the hEHR depends on the openness, trust and security of the architecture that will support the required functionalities. The next subsections will describe in detail how actors are expected to collaborate and how the citizen will manage their collaboration regarding his record.

3.1.2 Collaboration between Actors

Collaboration between actors is carried out by uploading contributions (Figure 3.2), and by accessing to a view of the information in the repository (Figure 3.3). Both actions are decided by the patient according to the Patient Access Policy.

In the process of creating information on a collaborative record, the different actors mentioned in Figure 3.1 can have a system that is able to generate a report based on OpenEHR archetypes or not. In cases where systems do not support them, external services with that functionality should be used. The archetypes for generation of the summary should also be available in a collaborative repository. After generation of the OpenEHR based report, it should be signed by the actor. Then he requests to submit the report to the patient repository. If the actor has submission privileges, the repository will store the report ending the process with success. Otherwise, the actor will be warned of the failure of the submission process based on the lack of privileges. The process of using the information from the repository is shown in Figure 3.3. External services for retrieving and displaying the information should exist for users who do not have a system that supports information retrieval in OpenEHR format. Systems that support it can access directly.

Whatever the method, the privileges of the requesting user are verified against the patient's access policy. If the actor cannot access the record, the procedure will warn the user of failure. If the user has privileges, then the system lets him choose if he wants to retrieve the full record or some subsets based on a query. All the sub-processes of querying and retrieving results

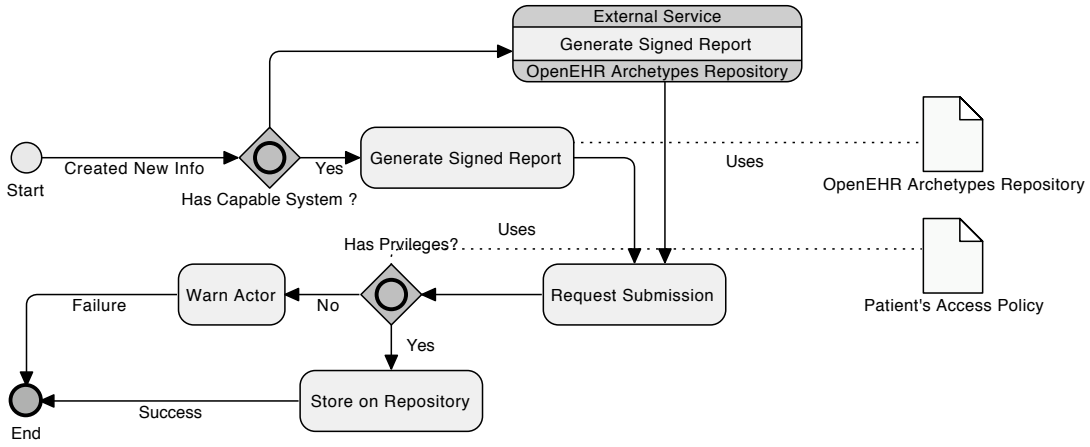

Figure 3.2: Actors Creating Information

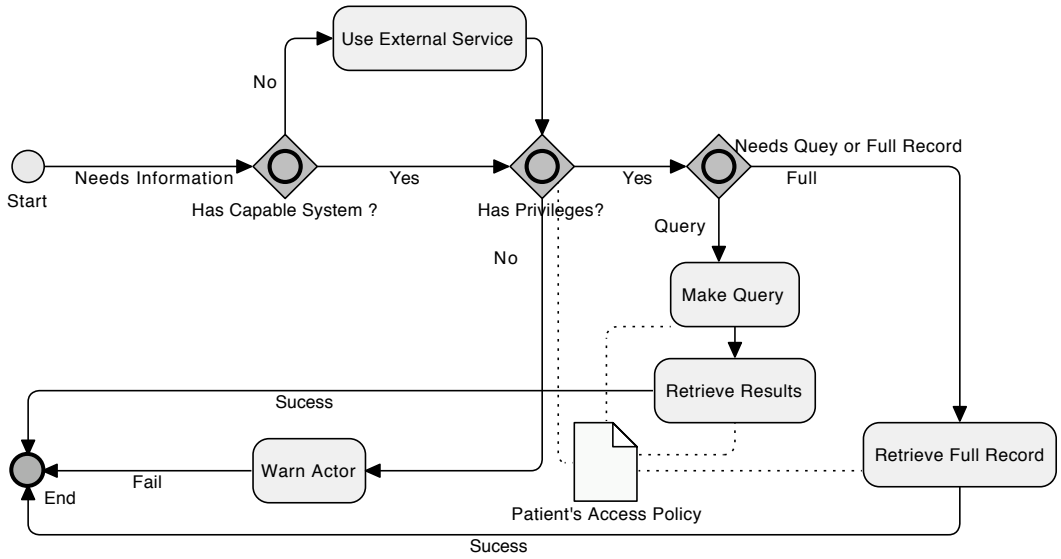

Figure 3.3: Actors Accessing Information

and retrieving the full record apply the patient's access policy, thus enabling more controlled access to the patient record and also restraining the actor from querying the full record. He will always work on the view granted by the access policy.

3.1.3 Managing Collaboration

To enable open collaboration, access is only dependent on the choice of the patient and the providing actor. Each new actor must self-enroll in the system by creating an account. This enables the system to authenticate providers and allows creation of a yellow pages service. The process is explained in Figure 3.4.

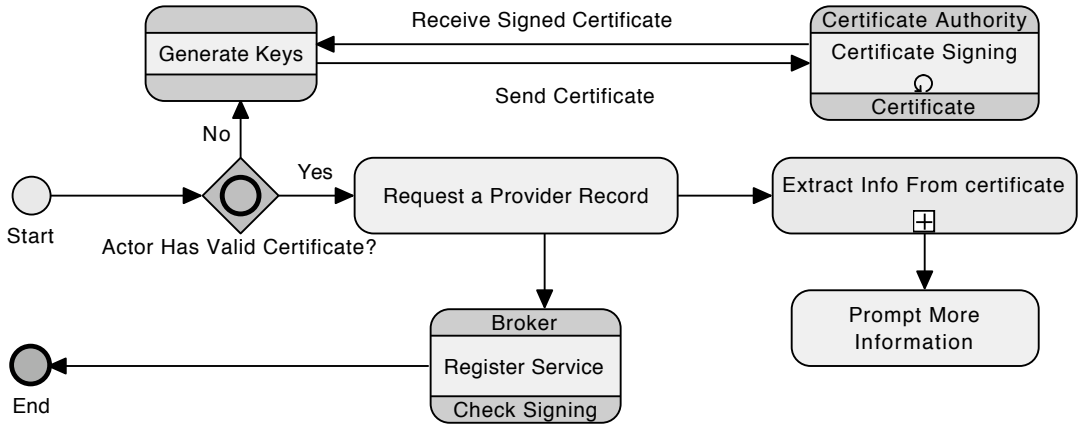

Figure 3.4: New Provider

To ensure trust in the system, the user should have a valid certificate, signed by a trusted external Certificate Authority (CA). If there is no valid certificate, the actor will generate a key pair and send his public-key with identification information to be validated by the CA. Then, the CA generates and signs a valid certificate for the actor. With a valid certificate, the actor must create and sign a service description which identifies the services that the actor provides along with an address and other important information. Then the actor should register the service description on the registry service, which validates the signing.

The patient should also create an account in order to gain access to the infrastructure. The process is shown in Figure 3.5.

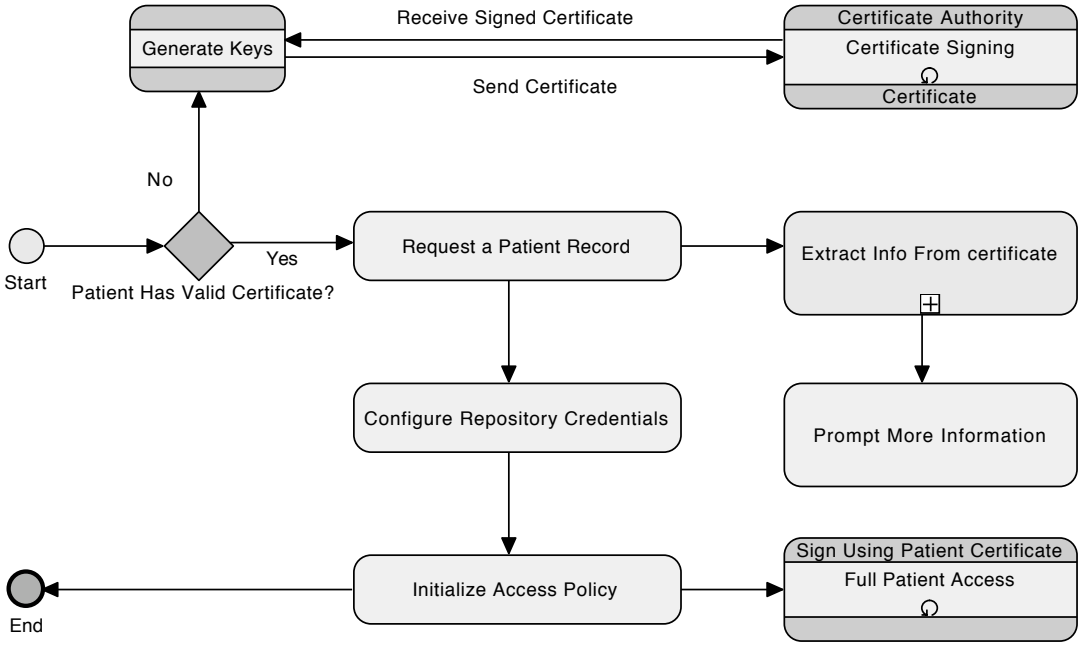

Figure 3.5: New Patient

If the patient does not have a valid certificate, he should ask for it to be generated and signed by a trusted external Certificate Authority (CA). The following step is to use that certificate to request a Virtual Health Card. First, the information needed to create the card is extracted from the certificate. That information will enable reliable authentication of the user and trust in his identity. Then the patient is asked for more personal information, still without privileges for changing information extracted from the certificate. Next, the repository associated with his record needs to be configured. In this step, the patient will provide information that is needed to gain access to the repository to be stored in his Virtual Health Card. Another important component of the Virtual Health Card is the access policy, and this policy will control access to the patient's Hybrid Electronic Health Record. The final step in creating the Virtual Health Card is initiating the policy. The policy is initiated to grant the patient full access to his data, followed by signing of the policy using the patient certificate.

The aforementioned process enables patients and providers to use the infrastructure. To allow collaboration between them, the patient should manage his policy, with the patient having full control over what information can be retrieved from his health record and who can retrieve it. One of the challenges is related to fine grain policies, because managing those rules is a complex and arduous procedure for the patient. Therefore, to enable the use of fine grain policies and reduce the patient's workload, the access control is based on functional roles extended from the standard EN 13606-4 [10], as presented in Subsection 2.5.2. Therefore, the patient chooses roles that are more suitable to the provider. Then, using the mapping between the functional roles and the sensitivity of the information, the system will grant access to the information. To extend the creation of more restrained access, the patient can create groups for aggregating providers, thus enabling collaboration between those providers in a more contained access. The access control mechanism will be described in more detail in the architecture section.

Figure 3.6 explains access policy management by the patient. The creation process is triggered either by the patient or by the provider. When triggered by the patient, he searches on the Broker for the Provider and gets the provider's identification. When triggered by the provider, the patient will receive a request from the provider and get the provider's identification. With the provider's identification, the patient can choose the roles of the provider or groups, or both. The functional roles are common to all patients, while the patient groups are private. In the next step, the patient chooses how long he wants to give access to the provider. Finally, the patient uses his certificate to sign the created policy to enable future validation.

The patient can also remove policy rules and list them. The removal process will not delete the policy, but will update the field that stores how long the policy is enabled to zero. The operation of updating a policy is not contemplated. This procedure is made in two atomic steps, first removing the old policy, using the previous present removal process, followed by the creation of a new one. For supporting the hEHR presented, an adequate architecture that copes with the requirements will presented in detail in the following section.

3.2 Architecture

This section describes an architecture that can support the previously presented requirements of the hEHR. The architecture enables openness and ensures secure collaboration con-

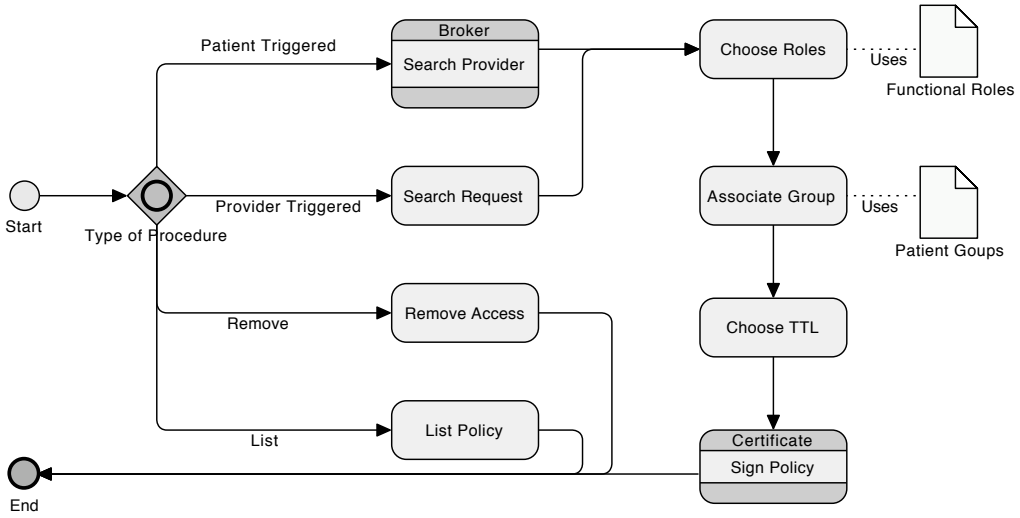

Figure 3.6: Managing the Access Policy

trolled by the patient. The solution mimics the behavior of a safe deposit box inside a bank and a proxy that will manage the information inside the safe deposit. The citizen will allow the proxy to deposit and withdraw information from the safe deposit to answer the needs of third parties previously approved by the citizen.

To ensure data protection, the safe will have small safes inside and the keys to them will be stored in another secure place. The information will not be visible to the bank. So if a requester needs to access the information, he will ask the citizen's proxy to provide the data. He will also give his representative an open safe that only the requester can open. The proxy, after consulting the policy for the requester will act accordingly. If accepted he goes to the bank and takes the small deposits that contain the information needed to the secure place with the keys to open the small safes. In that place, the citizen's safes are opened, information will be filtered according to the requester's policy and stored inside the requester's safe. The proxy will deliver the requester's safe to the requester himself. This approach means that the bank and the proxy cannot view inside the safes, the proxy can manage the safes but cannot open them. Also, the secure key deposit needs the proxy to gain access to the data as it cannot get the safe deposits. Figure 3.7 reproduces an overview of the actors and services of the architecture.

Using the previous analogy, the Virtual Health Card Service (VHCS)) is the secure key deposit, the Broker is the proxy and the Repository is the bank. Separation of the component that has the keys (the VHCS) and the one that has access to the Repository (the Broker) requires the agreement of two different components to give access consent. The Indexer will enable query and selective retrieval to or from the Repository. Also, an auxiliary service for accounting is used to store auditing information about requests to all the components of the architecture. The creation and use of information is through the Broker for all actors, patient and providers. The infrastructure deals with very critical information. Integrity and privacy are very important and to cope with these requirements the information is cyphered, as shown in Figure 3.8.

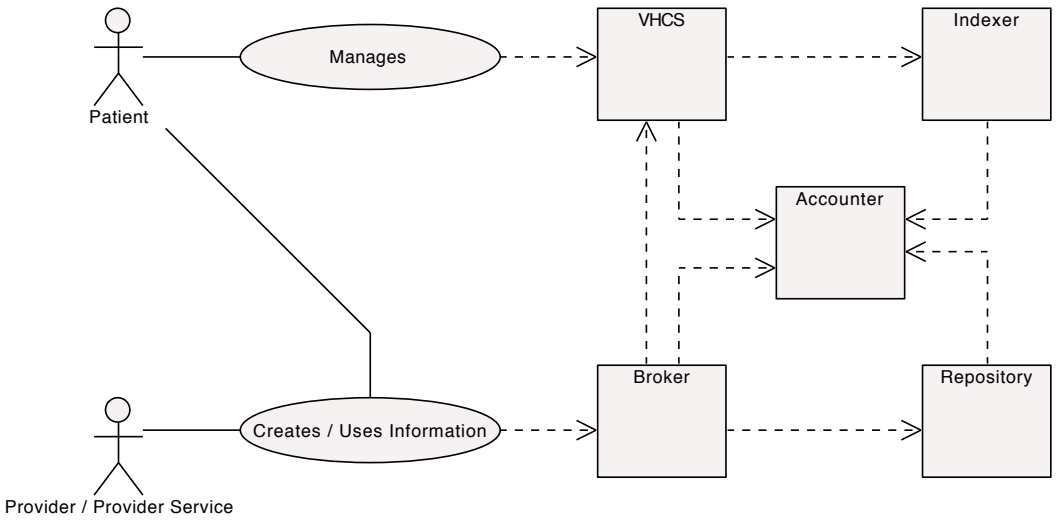

Figure 3.7: Architecture Overview

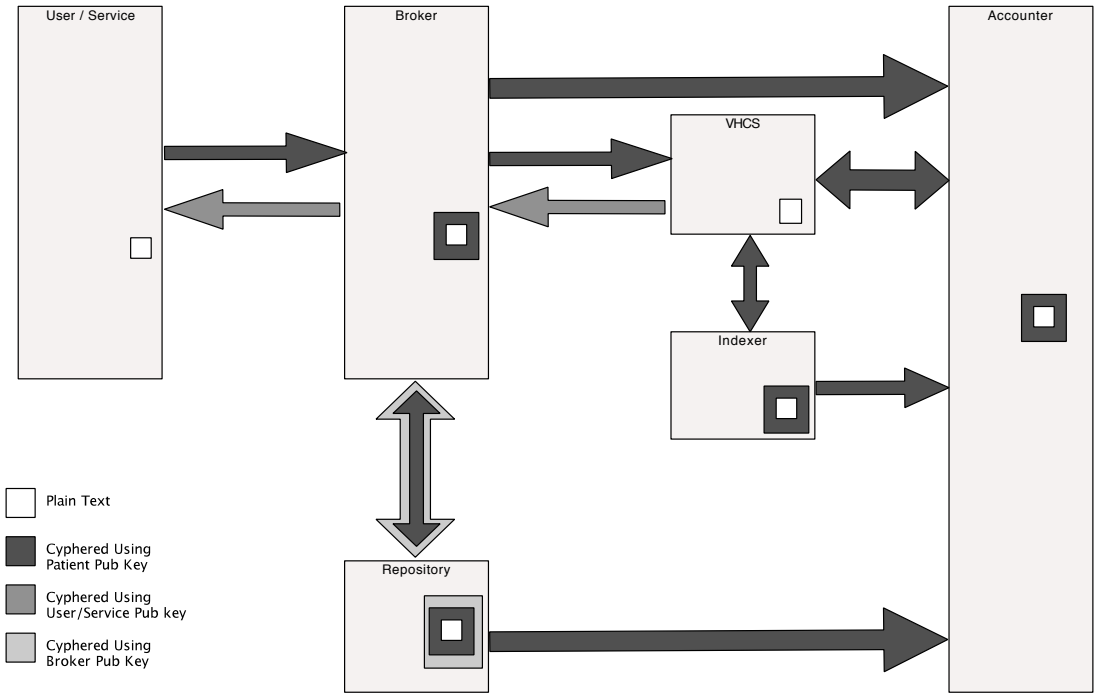

Figure 3.8: Communication and Data Store Privacy

The main idea is that information is patient-owned, so it is cyphered with the patient VHCS public-key. The patient record will be created by the contributions of each actor uploaded to the repository. In it, all contributions are stored individually cyphered to the patient public-key and signed by the producer. In the retrieval process those contributions will be reassembled to create a unique bulk of data. The information in transit is always

cyphered. Besides, two components of the VHCS manipulate the information in plain text, all other components using information in a closed envelope concept.

In the store procedure, the information is cyphered to the public-key of the patient before leaving user system. On retrieval, it is ciphered to the receiver before leaving the VHCS. The information related to indexing data of contributions deposited are also cyphered on the indexer (using the patient public-key) and the query matching will be made on the VHCS.

The Broker is the only service that can contact directly with the repository, acting as a middleman between the users and services that need access to the repository. It validates requesters and fulfils their requests using VHCS functionalities as needed. To ensure that the Broker is the component interacting with the Repository, bulks of data are also cyphered with the Broker's key. Therefore, the retrieval of information from the Repository requires the agreement of both the VHCS and the Broker.

The VHCS is responsible for storing the patient's credentials (public and private key), patient repository links, the correspondent repository credentials and the access control policy for the patient's information. VHCS provides functions and an interface that the Broker can request.

The Repository is a simple storage service that enables a service to request to store or retrieve a bulk of data associated with a specific identifier. The identifier is unique in each repository and associates the information stored with a specific patient.

All these components will push auditing logs to the Accounter component, thus enabling future checks and continuous auditing of services. The only component that can access those logs is the VHCS, because all the produced auditing will be cyphered with the patient public-key.

This approach will make great use of public-key cryptography. Therefore, the confidence between the components of the architecture and confidence between the patient and provider services will be ensured using Certificate Authorities.

The modeling of the architecture follows a Service Oriented Architecture (SOA), resorting to web services for the communication of components, thus making it suitable for future deployment in cloud-like services

Generally, management functions are carried out by the Patient using the VHCS, where he can manage repositories, his virtual health card and the access policy. The VHCS also has components to support the manipulation of patient data and support the other components.

The Indexer provides indexing capabilities for enabling query of data that is stored cyphered in the Repository. The Broker is responsible for managing the accounts for users who want to have access to patient information. It also works as a registry service, enabling patients to search for providers. It also provides request and store functions to every authorized actor.

The Repository has functions to allow the Broker to retrieve and store a specific contribution.

The Accounter provides Store Auditing Log to all components and Retrieve Auditing Log to the VHCS. In the next subsections, these components will be described in more detail.

3.2.1 Virtual Health Card Service

The purpose of the VHCS is to provide a service where the patient can store his credentials, which will be used inside the system, together with repository and access policies (Figure 3.9). This enables the disassociation of credentials that patients use to access the system from the

credentials used in the system. This will solve problems related to lost tokens, revalidation of credentials, and provide easier and secure procedures of information backup.

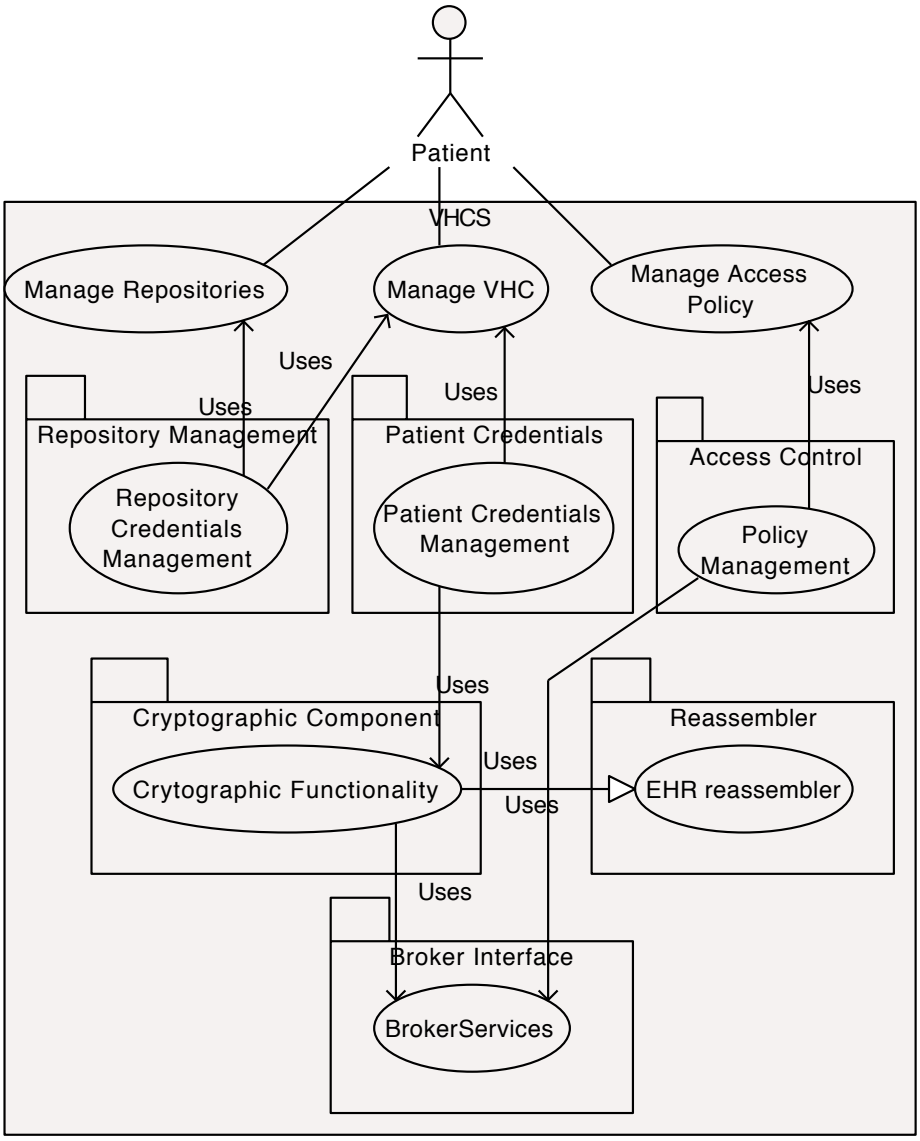

Figure 3.9: VHCS Overview

All the management information about a patient's health record is centralized, like the systems studied using smart cards. However, this approach allows services to access this information without the need for a physical card to be present. Moreover, this allows separation of the patient's demographic information from the EHR repository, because the Virtual Health Card will store the demographic information.

The VHCS is the only service that will be able to manipulate the information in plain text and take care of most of the cyphering process. Therefore, it provides more components than the ones used to manipulate the Virtual Health Card. It provides a cryptographic and signature validation of the information. Also, a reassembler component will be responsible

for gathering all the individual collaborations in just one bulk of data. Finally, it provides a broker interface to allow other services to communicate with the VHCS.

The service allows the disassociation of the credentials used in the architecture from the credentials used by the patient to authenticate himself in the architecture. Thus, it facilitates problems when the credentials used for patient authentication are compromised, as a new credential can be issued and associated with the correspondent Virtual Health Card. Also, when the internal credentials are compromised, the architecture can generate new ones and update the stored information to use the new credentials, without the need to re-issue a new certificate for the patient. Changes of credentials will be stored in a patient history log to enable future validation of data created or signed by revoked credentials.

One important aspect is trust and management of patient enrolment in the Virtual Health Card Service. The requirement of openness will foster the usage of a simple solution to self-enrol patients in the system. To establish trust, it is proposed that the patient will use his electronic identification card. Those cards are being currently implemented in various European countries and most have certificates signed by a government Certificate Authority. Therefore, the Virtual Health Card Service will rely on countries' Certificate Authorities to validate the certificates provided by patients and as such guarantee the identity of all patients. For situations when countries do not have a PKI in place, specific Certificate Authorities can be used. A new patient is registered by creating a new virtual health card, as follows:

- Authentication of the patient using his electronic identification card;
- Extraction of information from the card;
- Querying the user for complementary information:
  - Demographic Information;
  - Repository Information.
- Generation of the internal keys;
- Storing all the information;
- Initialize policy for enabling patient to access his record.

The service needs a database back-end to enable persistence of this information, and its structure is presented in Figure 3.10.

The database's back-end is used for storing demographics information about the patient, together with information for patient authentication, repository access and patient access policy. When necessary, this structure can evolve to store more information, especially demographic one in the demographic table. The Table tokens are responsible for aggregating information to associate the patient with the token used for authentication. The table *internal_tokens*, stores information and the internal tokens for the patient used in the architecture. The *repository* table stores information for access to the patient repository. The tables *groups*, *group_provider* and *policy* are used for patients' access policy management.

The authentication method is solid, since the electronic identification card asks the patient to insert the PIN to unlock the certificate to enable authentication. Then the challenge issued by the service verifies that if the certificate is valid, checking the signature of the trusted Certificate Authority. The unique identifier for each patient is the Distinguished Name (DN) of

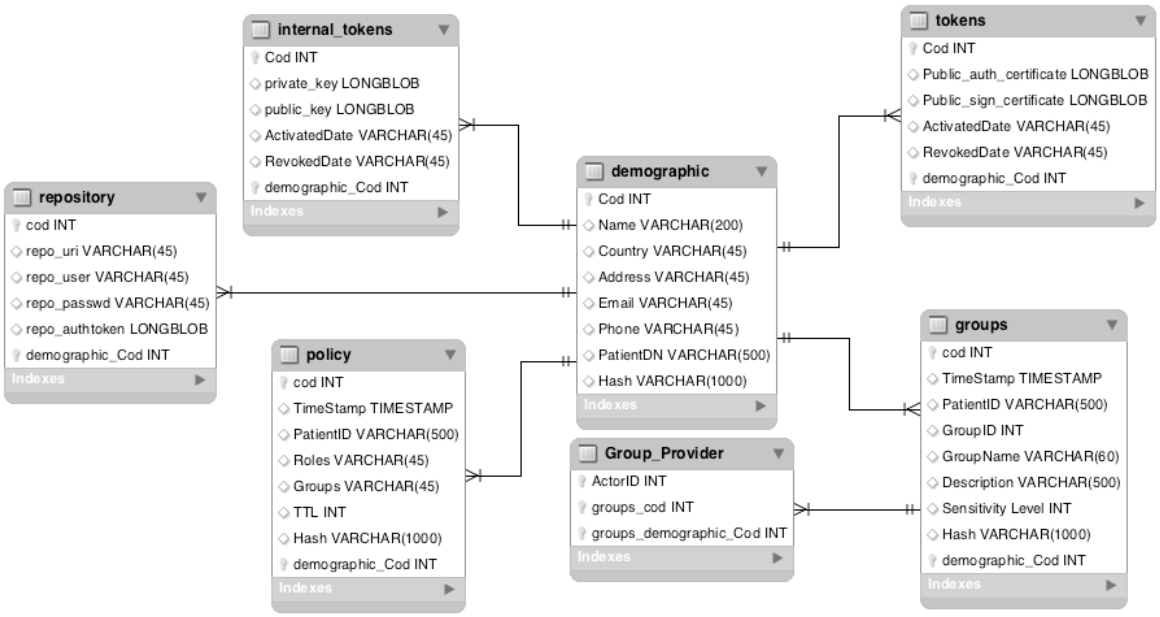

Figure 3.10: VHCS Data Model

the certificate. The structure of this DN can allow the system to check some of the information on the database. For instance, using the Portuguese citizen's card the DN has the following structure: *CN=Full Name, SERIALNUMBER=Identification Number, GIVENNAME=First Name, SURNAME=Surnames, OU= Cidadão Português, OU = Autenticação de Cidadão, O = Cartão de Cidadão, C=PT.*

This DN validates the citizen's identification number by checking the *SERIALNUMBER* field. The public keys are also extracted and stored on the database (on table tokens) to enable validation of signatures and to cypher information whenever the recipient is the patient. Then, demographic information is requested from the patient (demographic table), the requested information is concatenated and signed using the certificate. The result is stored on the Hash field of the demographic table. Then the private and public keys are created in order to be used in the architecture (stored in table internal_tokens). Those keys will be used for ciphering/deciphering the information that will be stored in the Repository. The cryptographic component is responsible for manipulating those keys securely and it ensures that the private key never leaves the VHCS.

After this comes the process of configuring the repository to be used for storing the health data. It will store a uniform resource identifier to request retrieving and storing of data from the repository and the credentials to access the repository, this information is stored in the repository table. The Repository component will be detailed in Subsection 3.2.4.

The tokens used for patient authentication and internal tokens used on the architecture will eventually need to be revoked and new ones need to be associated. Hence, the tables *internal_tokens* and *tokens* store two dates associated with the token activation date, field *ActivatedDate* and the revocation date on the *RevokedDate* field. Those fields will ensure the systems keeps the history of all the credentials associated with a patient over time thus enabling future validation of operations even after the token has been revoked.

Patient authentication is based on the token associated with the virtual health card. The

Algorithm used to validate patient authentication is described in Algorithm 1.

Algorithm 1 Patient Authentication

```
1: function PatientAuthentication                    ▷ Authenticates Patient Using DN form Token
2:     TokenDN ← ChallengePatient(Token)             ▷ Challenges the patient to authenticate with the token
3:     if TokenDN then
4:         TokenID ← Token.getID(TokenDN)            ▷ Gets the Token ID if token is activated and non revoked
5:         if TokenID then
           return Success
6:         else
           return Fail
7:         end if
8:     else
           return Fail
9:     end if
10: end function
```

After creating an account, the patient can authenticate using the token previously associated with the virtual health card. Normally, a PIN will be requested from the user to gain access to the internal token certificate, enabling a response to the server challenge. The certificate will be validated, checking if the Certification Authority is recognized by the architecture and the challenge is successful (line two of Algorithm 1). Then the system will check if that token is valid for that patient checking the association dates. First, it checks if the time-stamp of the authentication request is after the *ActivatedDate* and then if the token is still active (if there is not a revoke date, or the revoke date is later than the time-stamp of the request). If it is valid, the authentication method returns success, and if not, it fails.

The patient can decide the access privileges to his repository. Through the access control component, they store the policies the patient defines for other users and services. To enable easier and comprehensive management for the patient, the component uses role-based access control which also enables the use of dynamic groups, allowing the patient to control more specific scenarios of sharing.

The coexistence of clinical and non-clinical actors with access to the repository, combined with a common space for clinical and non-clinical information, brings challenges to the access control mechanisms. The idea is to make use of the standard EN 13606-4 [10] with some modifications. The clinical data will use the sensitivity level with the functional roles. A new role is proposed – uploader – to be associated with actors that can only upload information to the patient repository. The roles that the patient can optionally choose are the ones specified on the standard 13606-4 [10], extended with the uploader as in Table 3.1. The field Code will be used for correlating the role with the sensitivity level of the information.

This approach enables the patient to use a more granular control. The process is also facilitated by the association of roles that he understands. By using those roles the architecture maintains coherence with the information systems of the clinical providers. This means the information can be classified on the side of the provider by using the same classification of the information based on the sensitivity values of the standard, as explained in Table 3.2.

The information about roles, level of sensitivity and the matching privileges are common to all users and this is saved in the access control component. Moreover, each patient will define a policy that associates their providers with the roles that best cope with the type of healthcare service they provide.

The patient could also allow actors to collaborate in a more delimitated group. In this approach, the service allows the patient to manage groups by aggregating actors. Hence, this approach can complement the use of roles and cope with collaboration of non-clinical actors,

Table 3.1: Roles - Adapted [10]

Roles Code	Functional Role	Brief Description
7	Subject of care	Principal data subject of the EHR
6	Subject of care agent	e.g., parent, guardian, carer or other legal representative
5	Personal healthcare professional	Healthcare professionals that are closest to the patient
4	Privileged healthcare professional	Nominated by the subject of care OR nominated by the healthcare facility
3	Healthcare professional	Party involved in providing direct care to the patient
2	Health-related professional	Party indirectly involved in patient care, teaching, research
1	Administrator	Any other parties supporting service provision to the patient
0	Uploader	Actor that only creates information on the patient repository

Table 3.2: Values of Sensitivity for each Record Component [10]

Sensitivity level	Data Sensitivity value	Description of intended access
5	Personal care	Patient himself and one or two other people trusted by him
4	Privileged care	A very small group caring intimately care to the patient
3	Clinical care	Default for normal clinical care access
2	Clinical management	Wider range of personnel not all of whom are actively caring for the patient
1	Care management	Wide range of administrative staff

and complementary and alternative medicine providers. For each group the patient should define also the sensitivity level of the information that will be created by group members, storing it in the Sensitivity Level. It will also enable actors not belonging to the group to access the information if they have a role that has access to that sensitivity level.

Hence, each patient can define his groups using Table 3.3 and associate actors with groups and roles. An example of such an association is depicted in Table 3.4.

Table 3.3: Patient Groups

TimeStamp	PatientID	GroupID	GroupName	Description	Sensitivity Level	Hash
2011-05-22 14:42:11	Patient1DN	1	Sportactors	Group for actors related with sports care providing	3	9bd4ee14b f46005e542 44b7b03f6e 806a164f993
...	...	...	...	...	...	...

To allow patients to create groups, they need to store them in Table 3.3. In fact, this allows the collaboration of a group of actors, not based on their role but only considering if they belong to the sharing group. Each new group is represented by a new line inserted in the table. Each line is signed using the patient authentication token and the result is stored in the Hash field, therefore enabling future validation of the groups created. In this way, groups not created by the patient can be detected.

Table 3.4: Patient Policy

TimeStamp	PatientID	ActorID	Roles	Groups	TTL	Hash
2011-05-28 14:42:11	PatientDN1	ActorDN6	3	1	48	8efb19e557a357eaf 97c318f2b624279b4 e723ab
2011-05-28 14:45:20	PatientDN1	ActorDN73	5		-1	c231e1634cdd66545 5a75e99d926f11f54 7a26ba
2011-05-28 14:50:20	PatientDN1	ActorDN32	0		-1	676959920c5ba5529 fa58ff3af6cc38a73 0b1f6f
2011-05-30 11:35:12	PatientDN1	ActorDN24		1	-1	3cbb3028872dc1c5e b003a36a4a08a7f71 3e2f04

In Table 3.4, each patient can define the access policy for his records. Each policy entry is identified by the combination of the patient's distinguished name and the actor's distinguished name. The patient has the option of associating the actor with a specific role, based on Table 3.1 He can also decide to associate the actor with groups defined by him, in Table 3.3.

The patient can decide if the policy has a time limit (TTL, in hours) or if it is valid until he decides to revoke it (TTL=-1). Using the same principle of the patient groups table, the information created by the patient is signed. In the Hash field, the result of the signature of the inserted data is stored using the patient authentication token.

A proactive procedure can be executed to check the integrity of the information in those tables. The workflow for checking the correctness of the information is delineated in Algorithm 2. All the fields of data to be validated are concatenated except the Stored Hash.

Algorithm 2 Check Signature

```
1: function CheckSignature(DataToValidate, StoredHash, StoreTimeStamp)
                ▷ Validates the data inserted (DataToValidate), comparing the StoredHash and the StoreTimeStamp
2:     PatientDN ← ExtractPatientDN(DataToValidate)
3:     PubCert ← GetPubCert(PatientDN, StoreTimeStamp)
4:     Hash ← GenHash(DataToValidade, PubCert)
5:     if Hash == StoredHash then
         return Success
6:     else
         return Fail
7:     end if
8: end function
9: function GetPubCert(PatientDN, StoreTimeStamp)
            ▷ Get the public key associated to the private on the token used during the creation of the information
10:    CodPatient ← SelectCodPatientByDN(PatientDN)                          ▷ Uses Demographic table
11:    PubCert ← SelectPublicSignByCodANDTime(CodPatient, StoreTimeStamp)▷ Uses tokens, to retrieve the
   public key valid for the CodPatient where the StoreTimeStamp is between ActivatedDate and RevokedDate
       return PubCert
12: end function
```

The information in those tables is used to control access to the repository. To validate requests, the system uses Algorithm 3.

Algorithm 3 Check Policy

```
1: function CheckPolicy(actorDN, patientDN, operation, dataHeader)  ▷ Verifies access for an operation requested
   by an actor to patient data using information from its header
2:     groups ← GetGroupsForProvider(actorDN, patientDN)            ▷ Get list of actor's groups for the patient
3:     roles ← GetRolesForProvider(actorDN, patientDN)              ▷ Get list of actor's roles for the patient
4:     if operation == Store then
5:         if roles.code_id ≥ Uploader.Code_id then                 ▷ Using the roles of table 3.1
         return Grant
6:         else if actorDN == groups then                  ▷ If actor belongs to a patient group, on table 3.4
         return Grant
7:         else
         return Deny
8:         end if
9:     else if operation == Retrieve then
10:        sensitivity_level ← GetDataSensitivity(dataHeader)
11:        group_of_creator ← GetCreatorGroup(dataHeader)
12:        if roles.code ≥ sensitivity_level then    ▷ Or apply an query to the mapping table between sensitivity and
   roles
         return Grant
13:        else if group_of_creator == groups then    ▷ Check if requester belongs to the same patient group of the
   producer
         return Grant
14:        end if
15:    else
         return Deny
16:    end if
17: end function
```

The VHCS have three more components; the cryptographic component, the reassembler

and the broker interface.

The cryptography component performs all the encryption and decryption of the bulks of data. It means the private keys never leave the VHCS and the unprotected information is only temporarily processed in this component (and in the reassembler during the retrieval process).

The reassembler component provides a way to aggregate all individual contributions when a user asks to receive several or all contributions in the repository. During this phase, the access control policies can be applied to discard information that the requesting user does not have access to.

The broker interface component enables the Broker to request functions from the VHCS. It allows the Broker to check the access policy, to know what Repository to contact to receive or to store the information, the methods and the credentials. It also enables the Broker to request the patient's public key and request ciphering or deciphering of data through the Cryptography component using that key. Moreover, another important function of the Cryptographic component is the ability to receive ciphered data from the patient key and the receiving actor's public key, for deciphering the information and ciphering it to the receiver. Summarizing, the broker interface deals with requests from the Broker component, with the most important services provided being:

- CheckPolicy(PatientDN, ActorDN, Operation) This method is invoked when the broker needs to check if an Actor has access to perform the Operation (Store or Retrieve) requested for a specific Patient.

- GetPatientPubIntKey(PatientDN, ActorDN) This method is used when the broker needs to receive the associated patient Public Key to forward it to a specific Actor. This is used for enabling the actor to cipher the data to the patient before sending it to the broker and also for ciphering queries.

- GetRepositoryInfo(PatientDN, ActorDN) This method is used when the broker needs to gain access to a patient repository to perform actions requested by a specific actor. It will return the information necessary for enabling the broker to make use of the patient repository.

- CypherDataFor(Data,PatientDN,ActorToDN,ActorToPublicKey) This method is used when the Broker retrieves data from the repository and needs to reassemble it to send to a specific actor. The data is sent to the VHCS, which is able to decipher it using the patient internal private key. Then it reassembles all the bulks to which the Actor has access, and will cypher the result using the Actor Public key, send it back to the Broker to be forwarded to the requester.

- Query(CypheredQuery, PatientDN, ActorDN) This method is used when actors want to receive only data that match a specific query. The actor sends the Broker the cyphered query using the key returned by the GetPatientPubIntKey method. Then the Broker requests the VHCS to query a specific Patient, as explained in detail in Algorithm 4.

- AuthenticatePatient(PatientDN) This method is used when the Broker needs to authenticate a user that is a patient, requesting the VHCS to carry out authentication. If this is successful, the PatientDN is extracted and used for querying the VHCS if that token is valid.

- StoreIndexer(PatientDN, ActorID, BulkID, Sensitivity, CypheredHeader) This method is used when the Broker stores new information, depositing a cypheredHeader with information to be used for querying about a specific patient by an actor. It also receives the sensitivity level of the information, the bulkID in the repository for direct retrieval and the actorID of the creator of the information. Before invoking the method on the Indexer, Indexer.Store(PatientID, ActorID, BulkID, Sensitivity, CypheredHeader), the PatientDN is converted to his PatientID.

The query methods explained in Algorithm 4 enable the Broker to retrieve the bulkIDs from the Repository that match a specific query made by an actor.

Algorithm 4 VHCS Query

function QUERY(CypheredQuery, PatientDN, ActorID) ▷ Queries patient's repository metadata for an actor
    $patientID \leftarrow$ PATIENT.GETID($PatientDN$) ▷ Converts PatientDN to his ID
    $ActorsSameGroupsofRequester \leftarrow$ GETACTORONSAMEGROUPS($ActorID, PatientID$) ▷ Returns the ActorsIDs that belongs to the same groups that the ActorID
    $ActorRoleCode \leftarrow$ GETACTORROLECODEFORPATIENT($ActorID, PatientID$) ▷ Gets the code associated top the role that the patient defined to the Actor on the patient policy
    $temp_table \leftarrow$ INDEXERRETRIEVE($PatientID, ActorsSameGroupsofRequester, ActorRoleCode, ActorID$)
    $patientIntprivatekey \leftarrow$ GETPRIVATEINTKEY]($patientID$)
    $query \leftarrow$ DECYPHER($CypheredQuery, CypheredQuery$) ▷ Decipher the query using the patient internal private key
    $resultset \leftarrow$ SELECTBULKIDWHEREMETADATA($query$) ▷ Get the bulks identifiers where the metadata matches the actor query
    return results ▷ Returns the metadata to the Broker
end function

3.2.2 Broker

The Broker components are directly related to actions that other users or services perform over the patient repository. The main functions are associated with registry, storing and retrieving information, as shown in Figure 3.11.

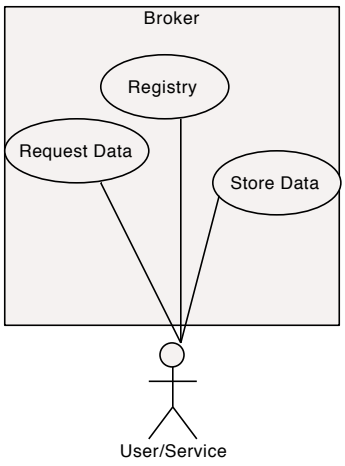

Figure 3.11: Broker Overview

The registry component is bound to user management. This component enables user registration and searching, and authenticates all actors that want access to a patient record,

even patient access through the Broker to store or retrieve information from his repository. The patient only accesses the VHCS for management purposes. When the patient accesses the Broker, the authentication process is forwarded to the VHCS. Therefore, the creation of a new patient and re-association of the patient's authentication token is made by the VHCS.

The creation of all other users is through self-enrolment on the Broker. This requires a certificate signed by a Certified Authority recognized by the architecture. Complementary information is requested from the user, thus enabling better identification and search capabilities for the patients. This allows the patient to search for a specific provider or a provider for a specific service.

The database that supports the Broker is shown in Figure 3.12.

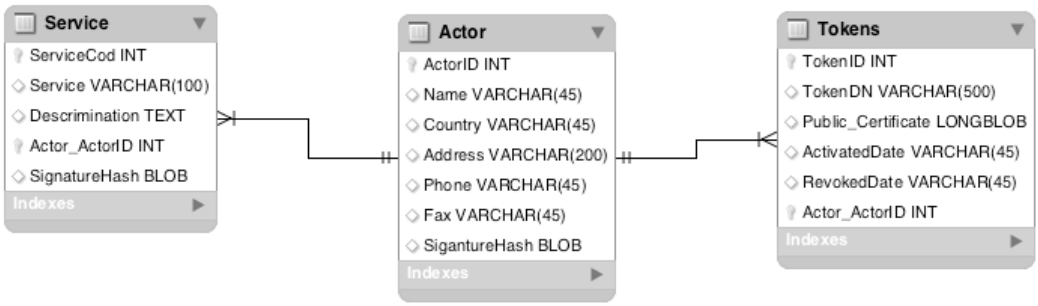

Figure 3.12: Broker Data Model

The Registry component stores the Distinguished Name and other information extracted from the certificate used by the provider for creation of the account. It also stores additional information requested from the provider, namely, address, contacts and service descriptions characterizing the available services on that provider. The information is signed using the provider certificate, thus enabling future validation of the stored information. The registration process is explained in Algorithm 5.

Algorithm 5 Registration

```
1: function RegisterActor(actorDN, actorInfo, actorSignature)        ▷ Register an Actor
2:     permission ← CheckValidToken()                                ▷ Checks if is a valid token
3:     if permission then
4:         creation ← CheckIfUserExist(actorDN)                      ▷ Checks if user already exist
5:         if !creation then                                         ▷ If not user create it
           StoreActor(actorDN, actorInfo, actorSignature)
           return True
6:         else                                                      ▷ Update Info
           UpdateActor(actorDN, actorInfo, actorSignature)
           return True
7:         end if
8:     else
           return Fail
9:     end if
10: end function
```

After registration, the provider can manage the services he provides. He is able to associate new ones or remove discontinued services. Thus, the patient can search for a specific provider or a provider of some service. One important aspect is how the Broker deals with the data using the closed envelope concept aforementioned.

After successful authentication, the actor can request the Broker to store or retrieve data. Algorithm 6 explains the storing procedure available in the Broker.

Algorithm 6 Store Data

```
1: function StoreBulk(actorDN, patientDN, datacyphered, dataHeadercyphered, sensitivity) ▷ Stores the data and dataHeader for a patientDN by an actorDN
2:     permission ← VHCS.CheckPolicy(actorDN, patientDN, "Store", NULL)        ▷ Checks privileges
3:     if permission = Grant then
4:         RepositoryInfo ← VHCS.GetRepositoryInfo(PatientDn, ActorDN)
5:         doublecyphered ← Cypher(datacyphered, BrokerKey)
6:         bulkid ← StoreOnRepository(doublecyphered, RepositoryInfo)
7:         patientPubkey ← VHCS.GetPatientPubIntKey(patientDN, actorDN)
8:         cypheredbulkid ← Cypher(bulkid, patientPubkey)
9:         VHCS.StoreIndexer(patientDN, actorDN, cypheredbulkid, sensitivity, datacyphered)
10:    else
           return Fail
11:    end if
12: end function
```

The store method receives the actorDN requesting the store action and the patientDN representing the owner of the information being stored. It also receives ciphered data (datacyphered) and the header (dataHeadercyphered). Both are cyphered with the patient public key before leaving the actor. The actor also sends the sensitivity level of the information being stored. The header has the sensitivity classification for the enclosed data along with information considered useful by the actor to enable a search related to the stored information.

To consume the information in the repository, the Broker enables a method for retrieving the data. The retrieval process can be divided in two types of access. The actor can request all the patient's record, or a specific part, both enforcing the access policy defined by the patient. As a consequence of the information being stored cyphered with its structure and to increase security, the query mechanisms are not provided directly by the repository. To retrieve a specific subset of the record, a query mechanism supported on an indexer service is provided in the architecture. This will be further explained in the Indexer(Subsection 3.2.3).

After successful authentication, the actor can request to retrieve the patient record or a subset as follows in Algorithm 7.

The method receives an actorDN representing the requesting actor, a patientDN which identifies the patient and a cyphered query. The query is cyphered by the requesting actor using the patient internal public key. Thus, if the user wants all the record the query is empty, if not it will send a specific query.

The first step of the algorithm is to check if the actor is allowed to request the retrieve operation. The request goes to the VHCS to make the query to the Indexer and returns the bulk identifiers of the contributions that matched the query. Then the Broker requests from VHCS the information needed to enable it to retrieve the bulks of data. That information has the credentials to request the operation from the repository, with the url for invoking the web service methods to retrieve the bulk of data. The Broker also receives additional information that is used when the Broker Key is rotated, i.e. a flag that describes if the data in the repository is still being updated. If this flag is active it will use the old Broker key, if not it will use the new one. This information supports the process of revoking credentials that will be discussed in Subsection 3.3.6.

For each bulk identifier, the Broker concatenates the web service methods to retrieve the bulk of data with the bulk identifier and requests the Repository for the correspondent data bulk, using the received credentials. Since in the store procedure the Broker has double

Algorithm 7 Retrieve Data

```
1: function RETRIEVE(actorDN, patientDN, cypheredquery) ▷ Retrieves data using a query to a patientDN record by an actorDN
2:     permission ← VHCS.CHECKPOLICY(actorDN, patientDN, "Retrieve", NULL)  ▷ Checks privileges
3:     if permission = Grant then
4:         bulks ← VHCS.QUERY(cypheredquery, patientDN, actorDN)
5:         RepositoryInfo ← VHCS.GETREPOSITORYCREDENTIALS(PatientDn, ActorDN)
6:         NewKey ← REPOSTITORYINFO.GETBROKERKEYINFO
7:         retrieved_data = newvector
8:         for each bulk in bulks do
9:             retrieve_url ← RepositoryInfo.RetrieveURL + bulk
10:            doublecyphered_bulk ← REPOSITORY.RETRIEVE(retrieve_url, RepositoryInfo.username, RepositoryInfo.password)
11:            if NewKey == Old then
12:                if bulkdata_cyphered_modificationdate < BrokerKey.CreationDate then
13:                    bulkdata_cyphered ← DECYPHER(doublecyphered_bulk, BrokerOldKey)
14:                else
15:                    bulkdata_cyphered ← DECYPHER(doublecyphered_bulk, BrokerKey)
16:                end if
17:            else
18:                bulkdata_cyphered ← DECYPHER(doublecyphered_bulk, BrokerKey)
19:            end if
20:            retrieved_data.add(bulkdata_cyphered)
21:        end for
22:        ActorToPublicKey ← GETACTORPUBLICKEY(actorDN)
23:        cypheredtoactor ← CYPHERDATAFOR(retrieved_data, patientDN, actorDN, ActorToPublicKey)
        return cypheredtoactor
24:    else
        return Fail
25:    end if
26: end function
```

cyphered data, now the Broker will decipher it to remove the outer envelope using his internal key. In fact, if the flag returned the old key, this means that not all the patient's information is cyphered with the new key. Therefore, it also has to check if the date of modifying the bulk retrieved is more recent than the creation of the key. If it is more recent, the Broker uses the old key. If one of the previous conditions is not verified, the Broker uses the current key. Finally, the resultant bulks are temporarily stored in a vector for further manipulation.

Once the previous process ends, the vector has the bulks of data that matched the query. As they are still cyphered with the patient internal public key, they cannot be manipulated by the Broker. Therefore, the Broker will send this vector to the VHCS requesting deciphering the current protection and ciphering again according to the requester. The Broker will send the data, the patientDN and the public key of the requesting actor. The VHCS will decipher the data bulk by bulk, and using a reassembler it will aggregate all the individual bulks in just one. Then the individual bulk is cyphered using the requester public key. These operations are performed inside the cryptographic and reassembler component of the VHCS. To conclude, the VHCS sends the cyphered bulk to the Broker, which forwards it to the requesting actor.

The Broker works strictly with the broker interface component on the VHCS to enable the storage and retrieval of information from the Repository. The Broker manipulates the information using a closed envelope concept as the information is always ciphered for the receiver, the patient or the requesting user or service. The Broker provides services for managing actors' registration, credential management, methods of updating actors' information and service descriptions. It also provides a yellow pages service that enables patients to search for providers.

3.2.3 Indexer

The choice of cyphering the data along with the structure improves privacy. Cyphering only the data enables the structure to be analysed, and infer about the data stored in it. One of the challenges of dealing with cyphered information is the overhead caused by cyphering and decyphering operations. The information is stored along with the structure in a cyphered bulk. Therefore, the repository can only send all the bulks or a specific bulk, because the repository only manipulates bulk identifiers and not field of the contributions. This limitation is due to the cyphering structure and data of the bulk, e.g. to prevent actors to request all bulks where some field has specific value or other type of query. To minimize the need of retrieving all the bulks and deciphering them to enable the retrieval of a subset of information from the repositories we propose an Indexer. This service will enable query capabilities over the cyphered data and structure.

The indexer service will work as an external repository, with metadata for indexing the information stored for each patient. The Indexer service is only accessible through the VHCS. Some of the fields are stored in plain text and others considered more critical are stored cyphered using the patient internal public key. The fields in plain text enable the execution of some queries on the Indexer, the other fields being made by the VHCS. Therefore, for a complete query from an actor, the query is made first on the Indexer to extract only the metadata the actor has access to. Once in the VHCS, after deciphering the metadata, the query is made against the metadata.

The fields on the Indexer are explained in Table 3.5.

Table 3.5: Indexer fields

Field	Description	Encrypted
ID	Incremental identifier for indexer field	No
PatientID	Internal code that represents a patient	No
ProducerID	Internal code that represent the producer of the data associated	No
Sensitivity	Sensitivity level of the information associated	No
Metadata	Metadata generated by the producer of the information associated	Yes
BulkID	Bulk identifier for the patient repository	Yes

The ID field is an auto increment field, PatientID is the patient's identifier that only the VHCS can match, the ProducerID is the actor's identifier than only the Broker can match, and the sensitivity field stores the sensitivity level attributed to the stored bulk of data. These fields are in plain text and enable the VHCS to retrieve only the metadata that a specific actor has for a specific patient. The Metadata and the BulkID is cyphered using the patient internal public key, ensuring that only the VHCS can manipulate this data.

The services provided by the Indexer are the store and retrieve procedure. The store procedure just receives all the fields for the table and stores them. The retrieve is in Algorithm 8.

The result is forwarded to the VHCS. Then, using the patient internal private key, the VHCS can match the requester query on the metadata and give the Broker the information about what bulks of data should be retrieved from the repository. This method is requested by the VHCS Query method.

Algorithm 8 Indexer Retrieve

```
1: function IndexerRetrieve(PatientID, ActorsSameGroupsofRequester, ActorRoleCode, ActorID) ▷ Retrieves the metadata to the VHCS
2:     condition ← PatientID==PatientID
                                                                    ▷ Get rows of specific patient, if
       AND (Sensitivity >= ActorRoleCode
                                    ▷ The actor role is equal or superior to the sensitivity level of the information
       OR ActorsSameGroupsofRequester.has(ProducerID))
                ▷ Or if the requester belongs to a patient groups common between the requester and the producer
3:     resultset ← SelectWhere(condition)
                                                          ▷ Get rows that match the condition
4:     if resultset then
         return resultset
5:     else
         return NULL
6:     end if
7: end function
```

3.2.4 Repository

Achieving a secure and open collaboration architecture lies also in the patient's freedom to choose his repository which has to implement two mandatory methods to enable the Broker to store and retrieve a bulk of data. As the data are bulks that combine the data and their structure in an glsxml format in a double-cyphered bulk, the repository can use different back-ends, such as file-system, relational database, no-sql databases among others.

The methods required in the interface with the Broker are:

- Store(AuthUser, AuthPassword, DataBulk) This method is used by the Broker to store information in the repository, by requesting the repository to store the DataBulk using the AuthUser and AuthPassword credentials. This method should return the identifier whom the repository assigned to the stored bulk. This identifier will be used by the Indexer.

- Retrieve(AuthUser, AuthPassword, BulkID) This method enables the Broker to retrieve a specific bulk of data from the repository, making use of the credentials for authenticating in the Repository – AuthUser and AuthPassword, and requests the data bulk identified by the BulkID.

If the Repository accepts the reception of extra parameters, such as the information of the ActorID that is requesting the information to the Broker, it will enable the Repository to log also information about which actor originated this operation over the Repository. Besides, the custom repository should make use of the Accounter store method to save its accounting information in an external service. To allow use of the accounter, the repository should provide a method for the patient to associate his internal public key with his account in the Repository.

3.2.5 Accounter

The Accounter is a service that will allow examination of the correct behaviour of the system, and if the policies are accurately enforced and respected by all actors.

All services will contribute to the Accounter by storing all the requests they receive, combining the granted and refused requests with the full request, identifying the requester, the time stamp and the operation. As this information can also reveal information that can

be used to extrapolate patient record pathologies and treatments, the information is stored cyphered using the patient internal public key. To maintain the integrity of the information, the services sign the information with their certificate prior to cyphering to the patient internal public key.

Considering the information stored in the Accounter, Table 3.6 details the information stored in plain text and cyphered.

Table 3.6: Accounter fields

Field	Description	Encrypted
ID	Incremental identifier for indexer field	No
TimeStamp	Time Stamp associated with the operation	No
TimeStampSignature	Hash result of signing the TimeStamp made by the Service	No
ServiceDN	Service Distinguished Name, provided from service certificate	No
ServiceDNSignature	Hash result of signing the ServiceDN made by the Service	No
PatientID	Internal code that represents a patient	No
PatientIDSignature	Hash result of signing the PatientID	No
ActorID	Internal code that represent the actor that generated the accounting info	No
ActorIDSignature	Hash result of signing the ActorID	No
AccountingInfo	Full description of the action requested, requester and exceptions generated	Yes

The plain text fields enable querying and retrieving more specific information from the Accounter. This allows a patient or patient services to retrieve all their accounting information, or anything particularly related to an actor or a service in a determined time frame.

The Accounter has two main methods that can be used remotely by web-services, the Store and the Retrieve. The Store method is available to all the services on the architecture, while the Retrieve can only be invoked by the VHCS. The Store method, depending on the service generating the accounting information, needs to convert some information to the correct form to be stored. Such operations are related to the local knowledge of each service and separation of information which prevents each one from having all the information. This operation is better explained by the algorithm used in the store procedure (Algorithm 9).

The information in the Accounter is produced by all the services in the architecture, with the exception of the Repository, which is external to the architecture. This means that all accountable services have to implement the procedure to store the events generated in each service in the Accounter. This procedure has some specific information, depending on the service generating the information. This method is executed whenever each service executes a request made to them. Generically, the procedure is explained in Algorithm 10.

The retrieve method provided by the Accounter, which is only available to the VHCS, can be used by patients to retrieve all their accounting information, or more specific information. Specific information can be queried using any combination possible using the fields that are in

Algorithm 9 Accounter Store

```
1: function STORE(TimeStamp, TimeStampSignature, ServiceDN, ServiceDNSignature, PatientInfo, PatientInfoSignature, ActorInfo, ActorInfoSignature, AccountingInfo)  ▷ Stores accounting info on Accounter
2:     if ServiceDN.has("Broker") then
                                                  ▷ Request VHCS to translate the PatientDN to PatientID
3:         PatientID, PatientIDSignature ← VHCS.GETPATIENTID(PatientInfo)
                                                  ▷ Broker knows the ID of the actor
4:         ActorID ← ActorInfo
5:         ActorIDSignature ← ActorInfoSignature
6:     else if ServiceDN.has("VHCS") then
                                                  ▷ Request Broker to translate ActorDN to ActorID
7:         ActorID, ActorIDSignature ← BROKER.GETACTORID(ActorInfo)
                                                  ▷ VHCS knows the ID of the patient
8:         PatientID ← PatientInfo
9:         PatientIDSignature ← PatientInfoSignature
10:    else if ServiceDN.has("Indexer") then
                                                  ▷ Indexer knows the ID of the patient and the actor
11:        PatientID ← PatientInfo
12:        PatientIDSignature ← PatientInfoSignature
13:        ActorID ← ActorInfo
14:        ActorIDSignature ← ActorInfoSignature
15:    else if ServiceDN.has("Repository") then
           ▷ Repository doesn't know the PatientID associated with the Repository, it will query it to the VHCS using the internal public key of the patient
16:        PatientID, PatientIDSignature ← VHCS.GETSIGNEDPATIENTIDBYINTPUBKEY(PatientInfo)
                                                  ▷ Repository knows the ID of the actor
17:        ActorID ← ActorInfo
18:        ActorIDSignature ← ActorInfoSignature
19:    end if
       STOREONACCOUNTER(TimeStamp, TimeStampSignature, ServiceDN, ServiceDNSignature, PatientID, PatientIDSignature, ActorID, ActorIDSignature, AccountingInfo)
20: end function
```

Algorithm 10 Accounter Store Method usage by the Services

```
1: function STORETOACCOUNTER                      ▷ Method executed in all architecture services
                                                  ▷ These procedures are common to all architecture serices
2:     TimeStamp ← TIMESTAMP()
3:     TimeStampSignature ← SIGN(TimeStamp, ServicePrivateKey)
4:     ServiceDN ← GETSERVICEDN
5:     ServiceDNSignature ← SIGN(ServiceDN, ServicePrivateKey)
6:     PatientInfo ← GETPATIENTINFO()
7:     PatientInfoSignature ← SIGN(PatientInfo, ServicePrivateKey)
8:     if Service == "Repository" then            ▷ The repository has locally stored the patient public key
9:         PatientPubKey ← GETPATIENTPUBKEY()
10:    else
                                                  ▷ The other services will request it to the VHCS
11:        PatientPubKey ← VHCS.GETPATIENTPUBKEY(PatientInfo)
12:    end if
13:    AccountingInfo ← GENERATEACCOUNTINGINFO
14:    AccountingInfoSignature ← SIGN(AccountingInfo, ServicePrivateKey)
15:    AccountingInfo ← CONCATENATE(AccountingInfo, AccountingInfoSignature)
16:    AccountingInfo ← CYPHER(AccountingInfo, PatientPubKey)
       ACCOUNTER.STORE(TimeStamp, TimeStampSignature, ServiceDN, ServiceDNSignature, PatientInfo, PatientInfoSignature, ActorInfo, ActorInfoSignature, AccountingInfo)
17: end function
```

plain text on the Accounter, namely TimeStamp, ServiceDN, and ActorID. The PatientDN is locked to the identifier of the patient who is requesting the information from the Accounter.

Concerning the common repositories that cannot collaborate with the Accounter service, the patient can still make use of the information on the Accounter generated by the Broker. He

can request accounting information related to requests made by the Broker to the Repository and with the Repository logs check if the is receiving requests not originated by the Broker.

The Accounter can be used not only to verify that all the actors are acting in accordance with patient policies, but also to verify the correct execution of the request and all architecture participants to achieve the contemplated results. This continuous auditing enables valuable feedback to the architecture about status and health that can allow correction of unexpected errors or malfunctions. The auditing can also identify problems with the policies defined by the patient and alert him to correct them.

This service can support future development of pro-active measures to detect and contain misbehaviour based on the information mining enabled by the Accounter.

3.2.6 Summary

The previous sections delineated the components, how they interact with users and provide services between all of them. It was also important to explain how trust between the components is managed. The services of the architecture enable each component to make use of the services provided by another. It was explained how authentication is performed and how some functions are invoked, but before this procedure, the service itself is validated. This allows only trusted components to interact. The assumption is that a PKI for the architecture components exists, so each component has a certificate to enable correct identification and authentication between services, providing yet cryptographic capabilities.

The interaction between all the components, together with their functions, is illustrated in Figure 3.13. Besides those already listed, the OpenEHR archetype is present and a trusted third party for managing the Certificate Authority for the architecture components.

The PKI for components is directly related to trust between the architecture components. Each service requests a certificate from the PKI services, to use for authentication between each other through mutual certificate authentication. During authentication, the services verify the Certificate Revocation List of the PKI to check if the certificate is valid. The PKI also maintains a historic list of all certificates that were issued to the services, to be able to verify past log actions that were logged in the Accounter.

The OpenEHR archetype repository is used for storing and enabling retrieval by all actors of the archetypes used in the contributions stored in the patient repositories. This is also a collaborative repository that allows actors to develop new archetypes for dealing with new types of information that can be produced in new services. This will enable continuous evolution of the type of information that the records can manage.

Considering this approach, patients can manage, store and retrieve data from his own record using services from the architecture. Users or services that want to collaborate can also access to the services of the architecture. Already deployed systems can collaborate by creating a connector service that act as a user or service. Enabling Hospital Information Systems, national EHRs, epSoS or others to collaborate, as depicted in Figure 3.14.

3.3 Common Operations

Following the proposed architecture and behavior already described, this section explains how components interact to achieve the most important use cases: how to create a new patient or a new provider; how do the patient manage the access policies to his record; how can actors contribute to each EHR, namely storage and retrieval methods; and how new services can be

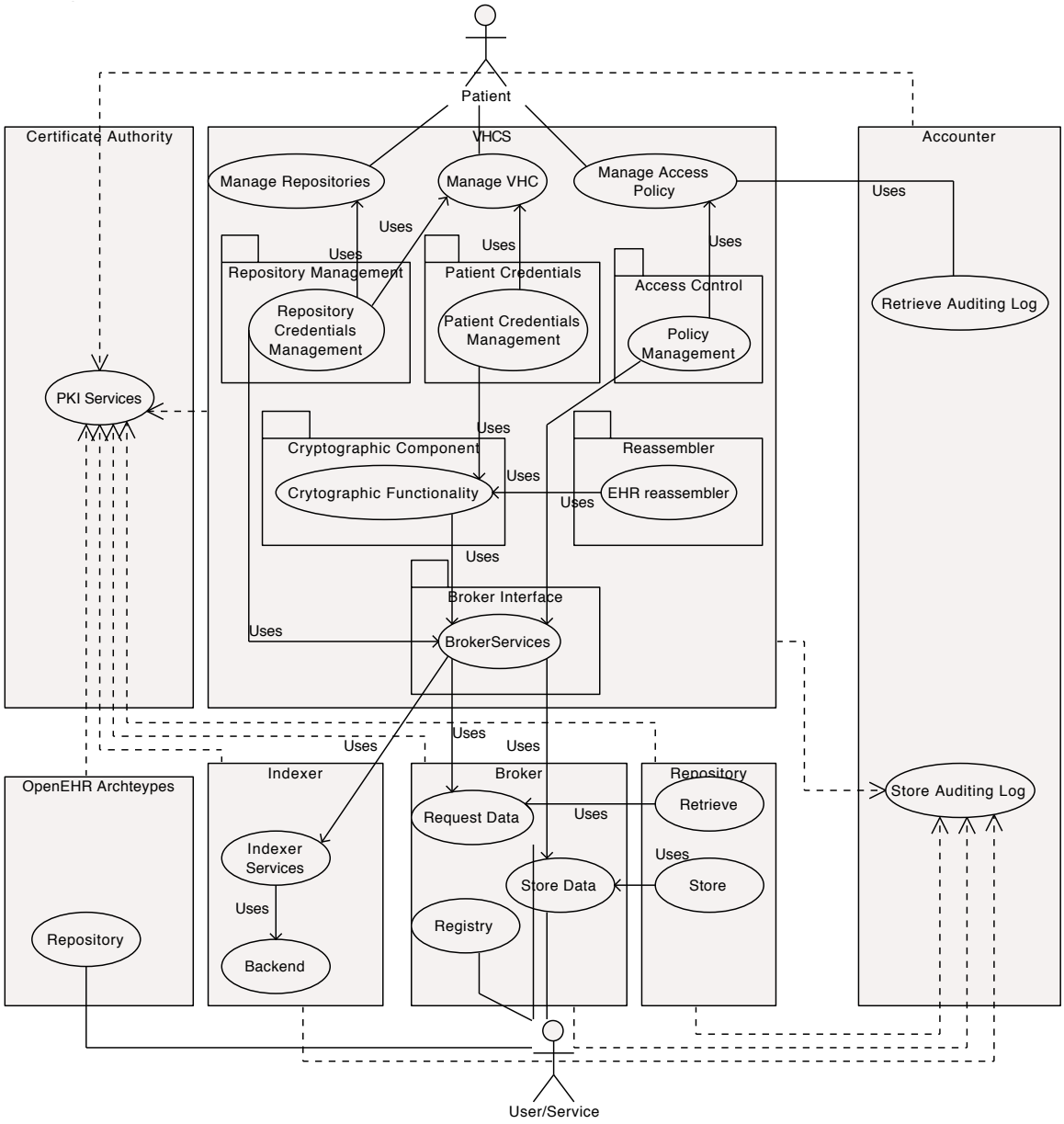

Figure 3.13: Interaction between users and services components

added to this core architecture. Moreover, credential revoking and digital signature validation are also addressed, and, finally, availability challenges are discussed at the end of the section.

3.3.1 New Patient

Figure 3.15 represents the states and interactions between services to create a new patient in the architecture.

Operations related to patient management are carried out through the VHCS. This process also needs the Accounter and Repository services. The Accounter stores accounting information while a new patient account is created. The Repository will only be used to

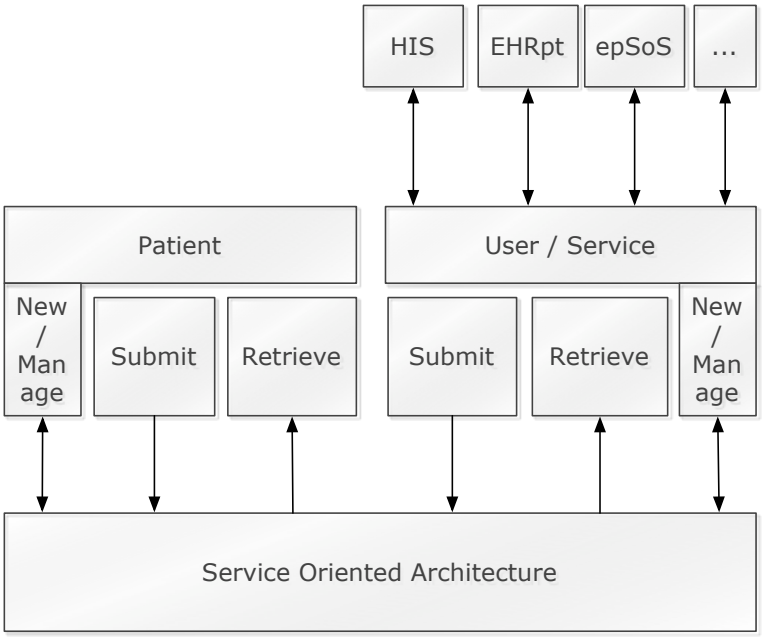

Figure 3.14: Connecting with existent repositories

verify the accuracy of the repository credentials and configuration.

The patient needs a certificate issued by a Certificate Authority trusted by the architecture and therefore by the VHCS. As countries are promoting the use of electronic identification cards, their Certificate Authorities are trusted by the system. Therefore, any citizen with a trusted signed certificate can self-enroll and create his account. Another prerequisite is to have an account in a repository that will enable storage and retrieval of data.

The process is simple. In step one the patient requests the creation of a new account to the VHCS. The VHCS carries out authentication using the certificate challenge and logs this information with the Accounter. The VHCS extracts the information from the certificate, using it to create the Virtual Electronic Card (step 4), requests more information from the patient that is not available on the certificate (step 6) and all phases are logged with the Accounter. Another request to the patient follows, for information related to the patient repository. Once this is received, the VHCS tries to connect to the repository to check if the credentials are correct. Finally, the VHCS stores all the information on the database. After this process the patient can connect to the VHCS and manage his Virtual Health Card. The logging messages sent to the Accounter are signed and cyphered using Algorithm 10 described earlier.

3.3.2 New Provider

The Broker is the service responsible for managing the actors in the architecture. So a provider will be created through interaction between the actor and the Broker service. The Accounter service is used to store all the logging information produced during the creation process. The logging information is sent to the Accounter using Algorithm 10.

Creation of a new provider is illustrated in Figure 3.16.

Responsibility for creating certificates for providers is delegated to Certificate Authorities

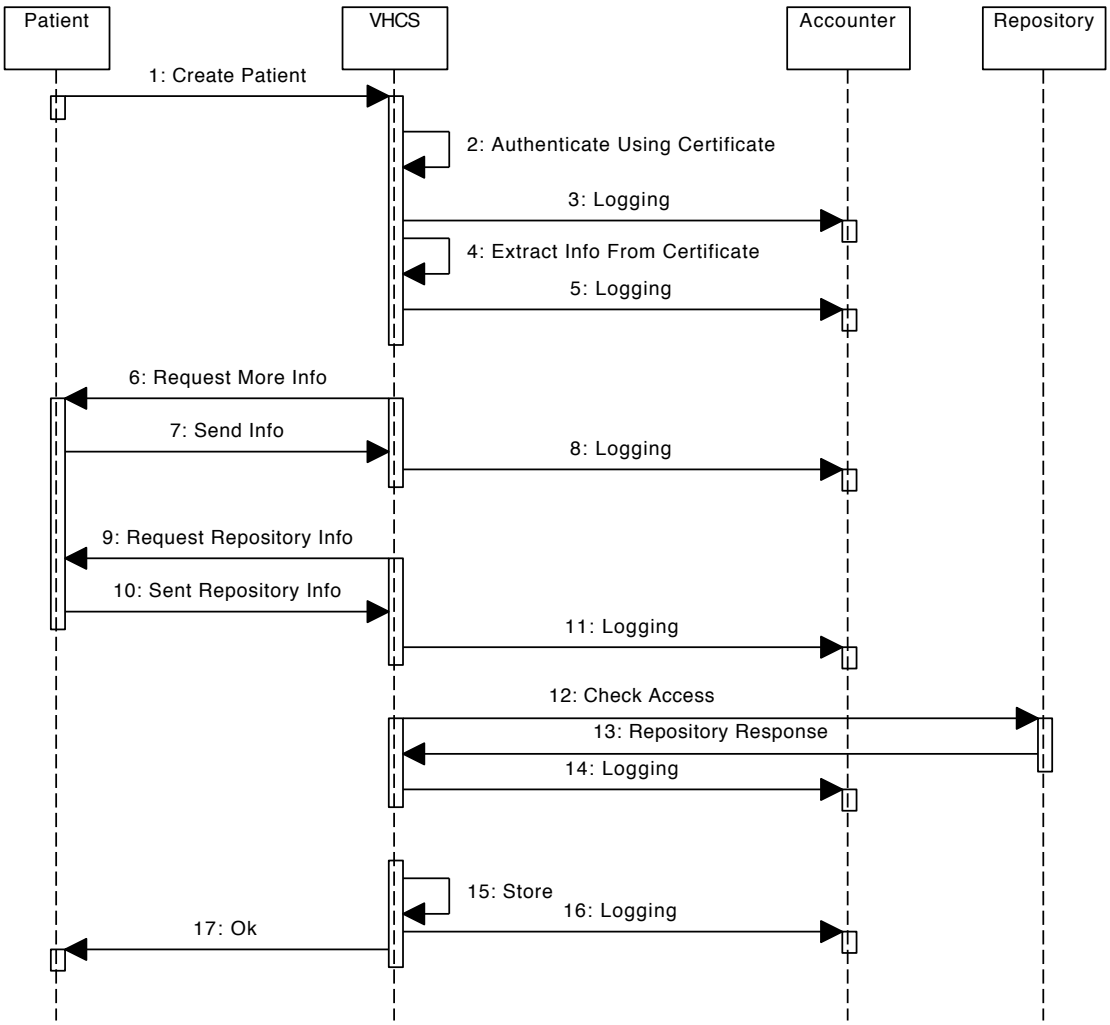

Figure 3.15: Creating new Patient

considered trustworthy in the country where the provider is registered. Therefore, the Broker requests the Actor to use his certificate authentication (step 2). The actor can progress to creating his account if the request is successful and the Certificate Authority is trusted.

The provider has to give more information related to their office, such as address, phone, other contact and descriptive information to create a complete entry. This information will also be used to enable a Yellow Pages Service on the Broker, enabling patients to search for providers. This information is signed using the provider authentication key, and then the information and the signature hash are sent to the Broker.

The actor also describes the services he has available to the patient, signing this information. Each piece of service information is signed by the actor and sent to the Broker.

When the actor stops sending the services he provides to the Broker, the Broker stores all the information, logs all the operations made and informs the actor of the success of the operation.

The information stored in the Broker enables providers to authenticate, and if the patient policy lets them, collaborate in creating or consuming information from the patient repository.

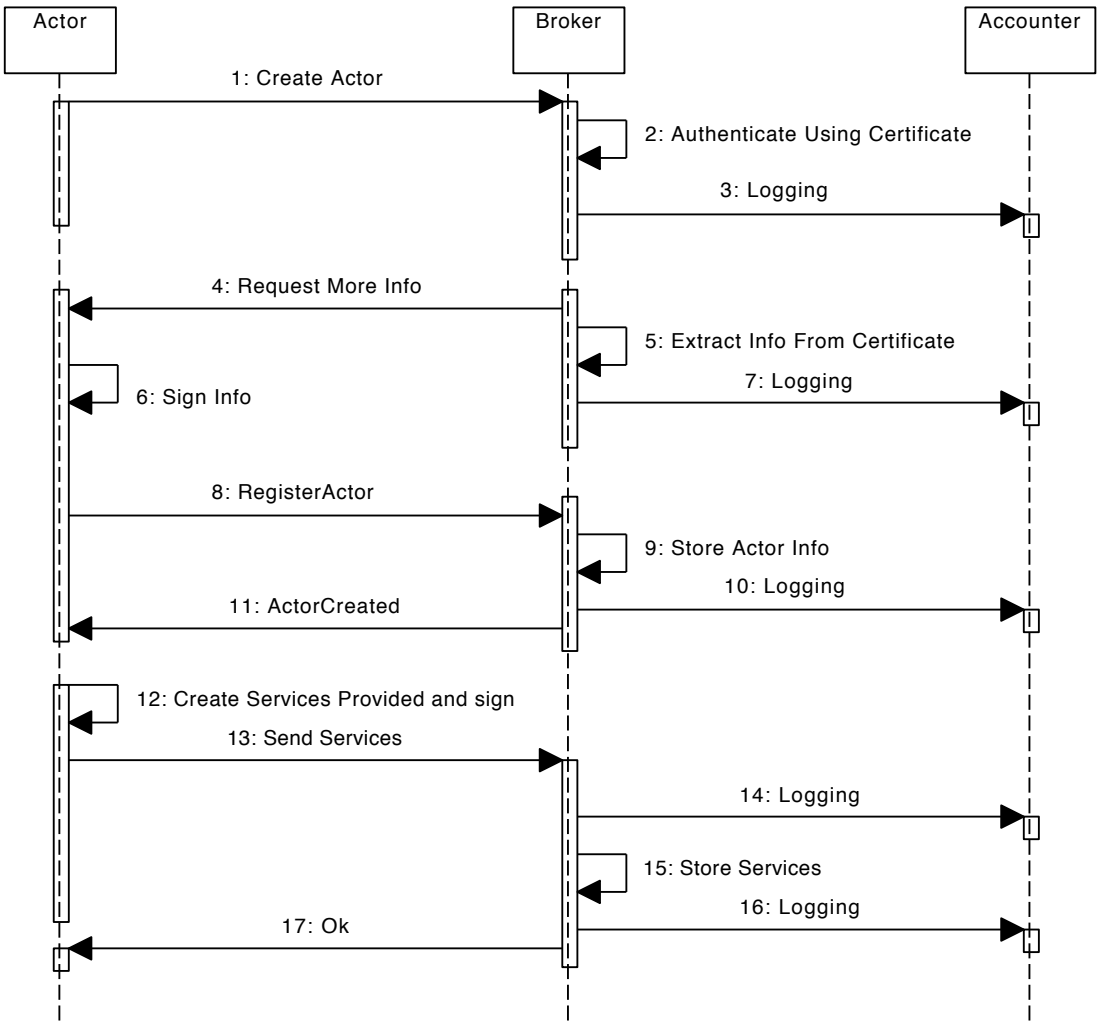

Figure 3.16: Creating new Provider

This also enables patients to search for a specific provider, or search for providers of a specific service in a specific area.

3.3.3 Managing Access Policies

Each patient can manage his policy, resorting to the services provided by the VHCS. The policy management component enables the patient to list, update, disable and create policies associated to each provider or group of providers.

The process of creating a new policy is explained in Figure 3.17.

The patient asks the VHCS to create a new policy for a specific provider, identified by his Distinguished Name. After authentication, the VHCS (step 2) retrieves from the Broker information related to the provider, forwarding it to the Patient. The patient can associate the provider with previously created groups, retrieving the group id from the VHCS (step 9). The patient can then choose the role he wants for this provider, using the role id. The patient also decides the TTL of the policy in hours. The value -1 leaves the policy active

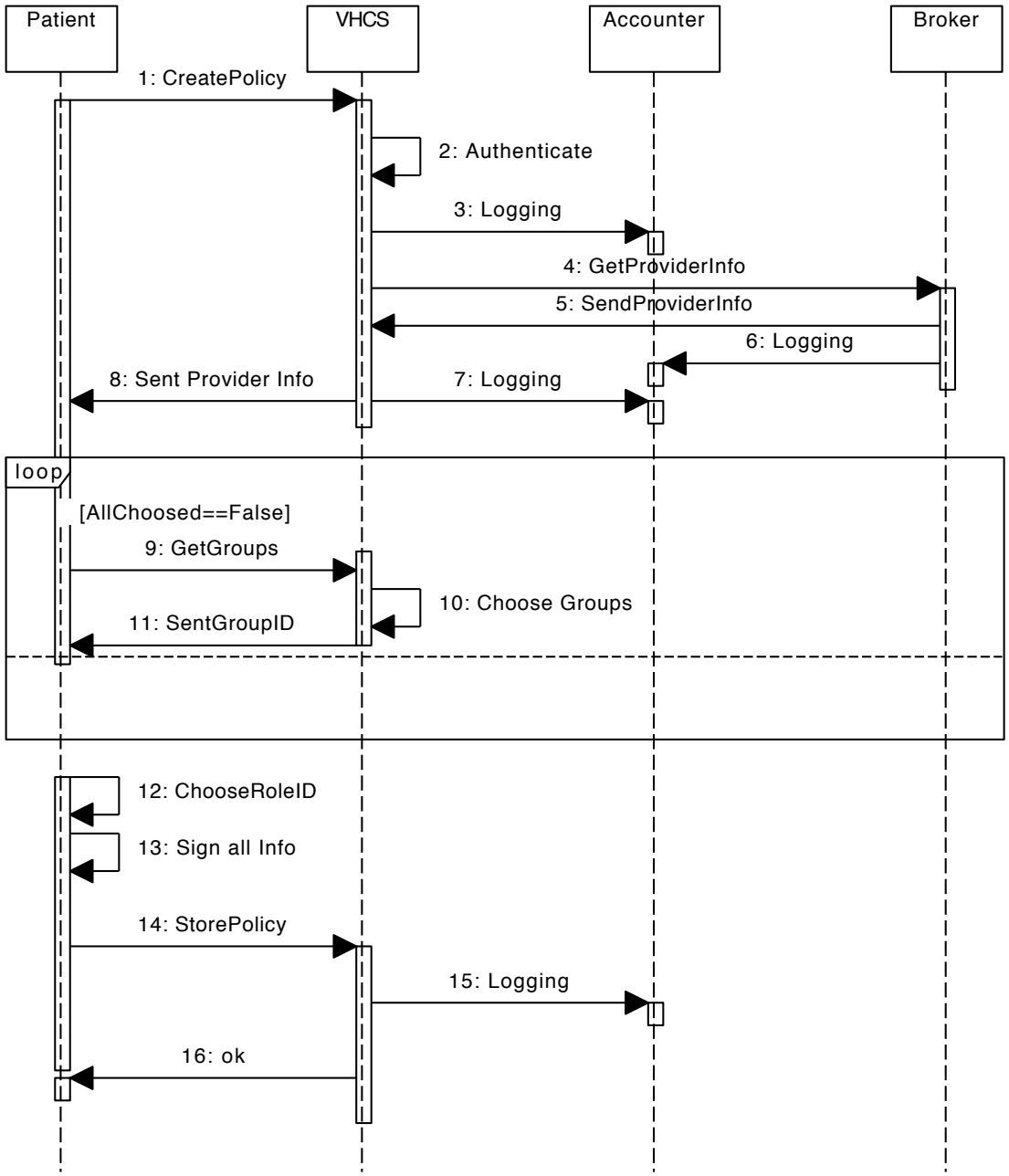

Figure 3.17: Creating a new policy

---

until the patient revoke it. Then all this information is signed, the resulting hash with this information is passed to the VHCS as arguments of the StorePolicy method (step 13). After correct storage, the VHCS logs the operations in the Accounter and sends an *ok* message to the Patient.

3.3.4 Storage

The storage process is requested by an actor through the Broker, using services provided by the VHCS, Repository, Indexer, Accounter, Certificate Authority of the architecture and the OpenEHR Archetypes repository. The interaction between all those components is described in Figure 3.18.

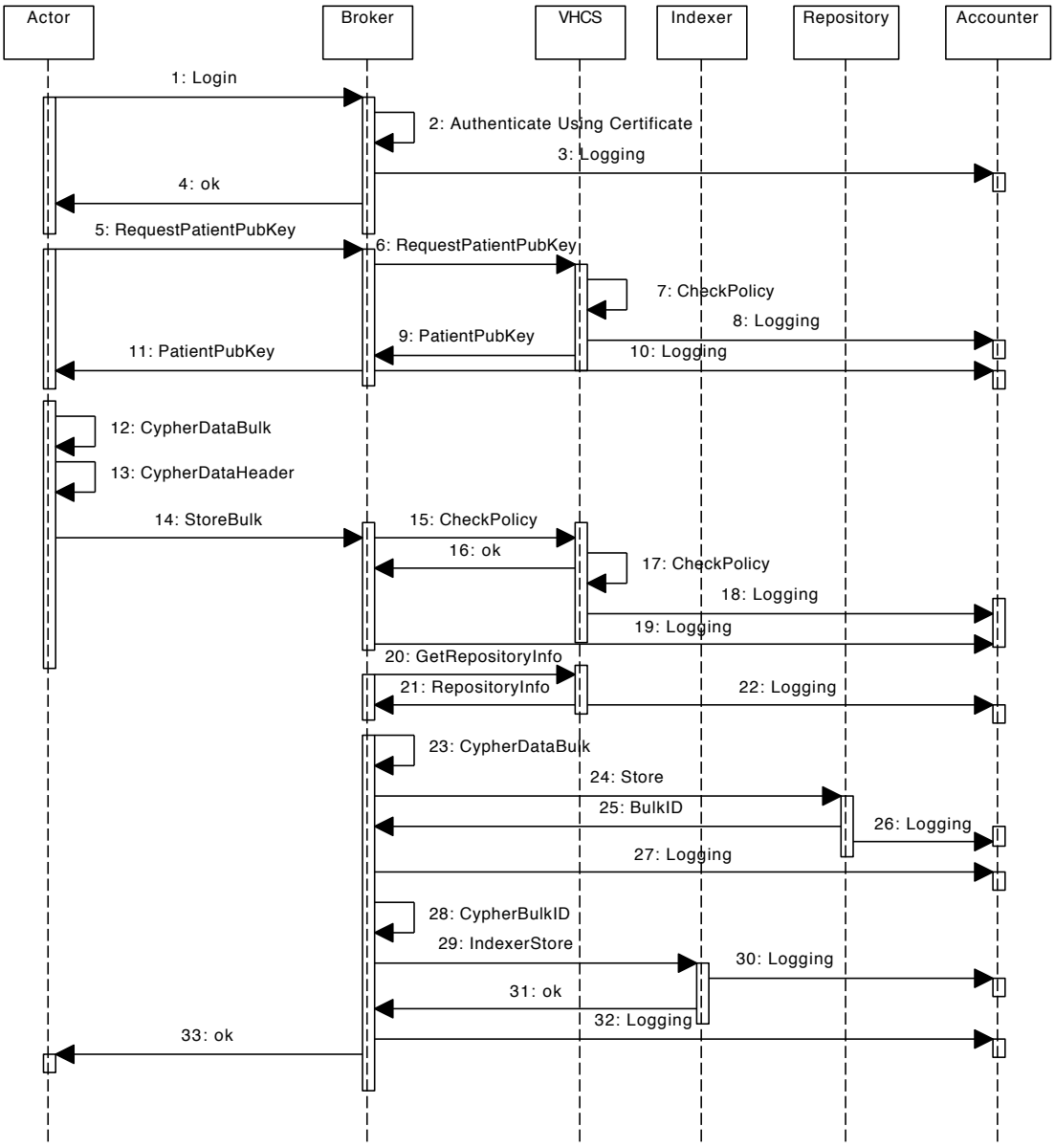

Figure 3.18: Storing a contribution

The actor uses his certificate to authenticate with the Broker. If successful, the actor requests the patient public key from the Broker, which forwards this request to the VHCS, which in turn returns the response to the Actor. With this information, the actor cyphers the bulk of data and data header. Then the actor asks the Broker to store (step 14), sending

the actor and patient Distinguished Name with the data, data header cyphered to the patient public key and the sensitivity level associated with the data.

With this information, the Broker checks if the policy allows the requesting actor to store information. If the actor has authorization, the Broker asks the VHCS for the Repository information of the patient. Then the Broker double cyphers the data information with his internal key and stores the data in the Repository using the information received from the VHCS (step 24). The Repository returns the identifier (ID) given to the new stored data bulk. The Broker then cyphers the ID using the patient public key. The Broker stores information on the Indexer, sending the patient identifier, actor identifier, sensitivity of data and the cyphered fields with the bulk identifier and data headers with the metadata. Once this is done, the Indexer sends an *ok* message to the Broker informing the actor that the store procedure occurred without problems. All the requests between services are registered in the Accounter service.

3.3.5 Retrieval

The retrieval process is shown in Figure 3.19. After the Actor authenticates using his certificate on the Broker, the Actor will request the patient's public key identified by the Patient's Distinguished Name that the Actor knows. The Broker will forward this request to the VHCS, which after checking the policy, returns the public key back to the Broker, which delivers it to the Actor. Then the latter can encrypt a query, enabling him to retrieve the data he needs from the repository.

The Actor requests the Broker to retrieve the data, sending the patient and his Distinguished Names with the cyphered query. Then the Broker requests the VHCS to send the repository information related to the patient. This information will enable the Broker to access the patient Repository.

Then the Broker forwards the query to the VHCS, which will answer with the bulks of information matching the query in the following steps. First, the VHCS checks that the requesting actor can make the query, then deciphers the query. Second, the VHCS retrieves from the Indexer the information that is needed to match the received query. Finally, the VHCS deciphers the metadata and the bulks' identification fields, executes the query and returns the result set (bulk identification) to the Broker.

The Broker can then retrieve each bulk from the repository and decipher it using the broker's secret key. Thus, the Broker requests the VHCS to cypher all the bulks to the Actor requesting the information. The VHCS deciphers using the patient internal private key, checks the actor's privileges and the granted bulks go to the reassembler to be consolidated as just one bulk. The resulting bulk is cyphered with the actor's public key. Then the cyphered data is forwarded to the Broker, which delivers it to the Actor.

During this process, all services log information to the Accounter, allowing accounting information about all processes for future validation of correctness and adherence to patient policies.

One important aspect that has not yet been discussed is the mutual authentication between services. Actors and patients use their certificates to authenticate with the Broker and the VHCS. But for a more secure environment, services may use mutual authentication using their certificates and then all requests between services and components are preceded by mutual authentication.

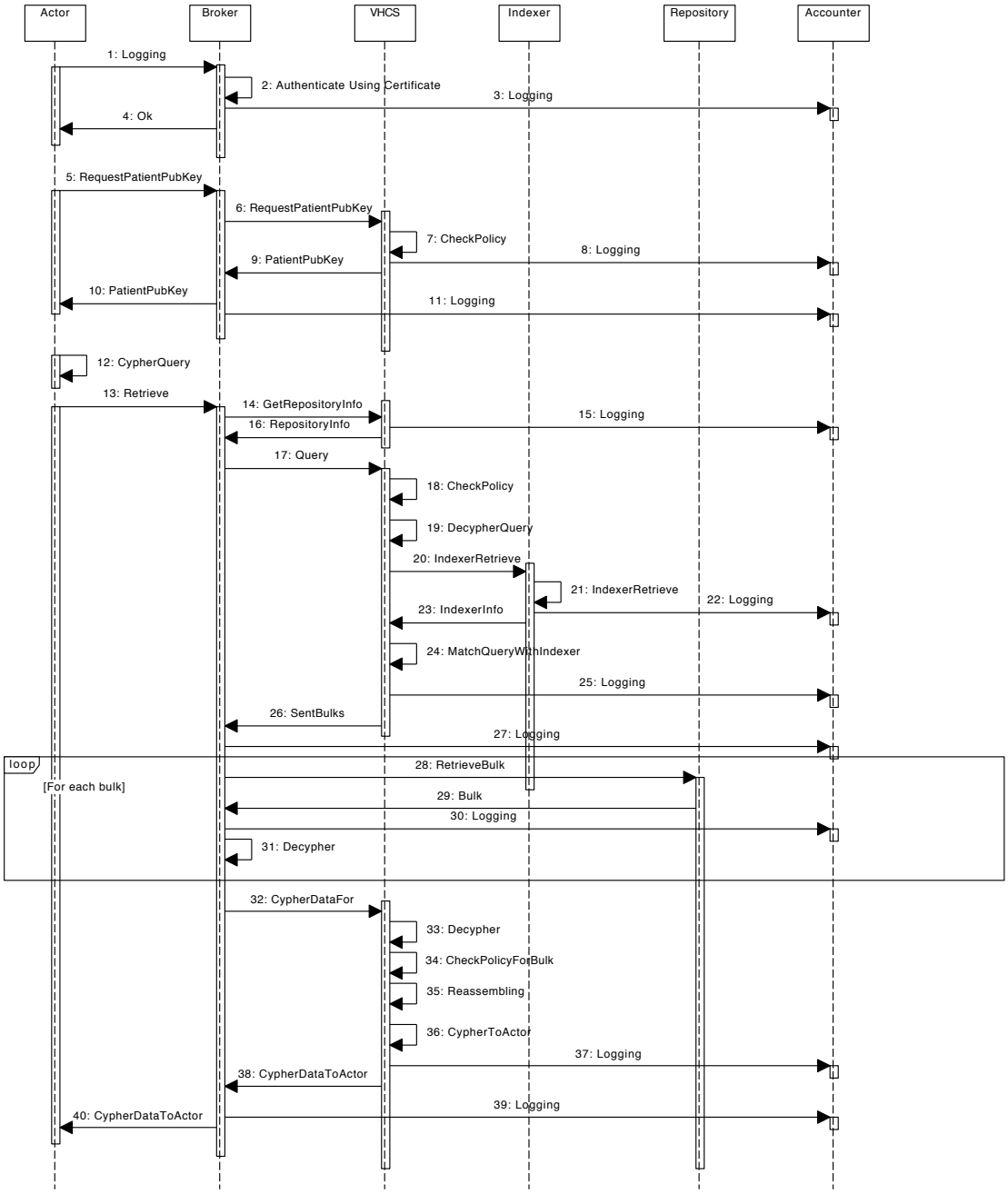

Figure 3.19: Retrieving data from repository

3.3.6 Revoking Credentials

Possibly, some credentials will be compromised. It may be an actor credential, a patient authentication token or internal patient keys. It can also be service credentials or services' internal keys.

If a patient has his authentication token compromised, he needs to ask his Certificate Authority to revoke the current certificate and issue a new one. Then he needs to associate

the new one with his account on the VHCS. This process is explained in Figure 3.20.

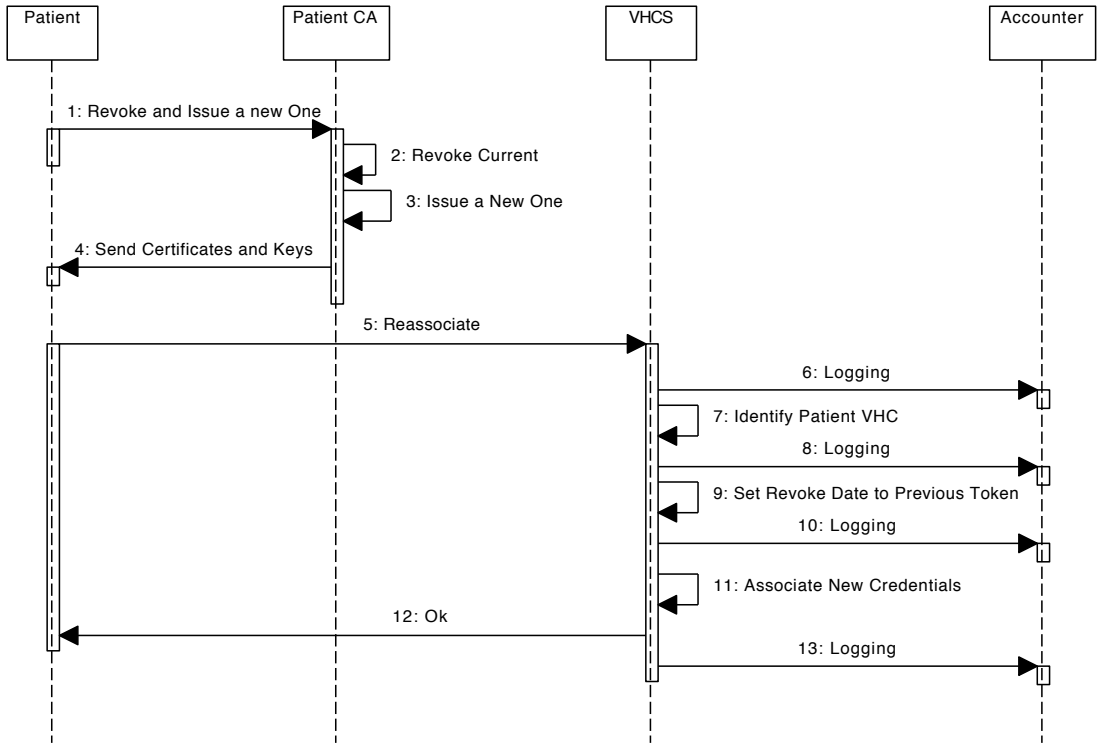

Figure 3.20: Revoking Patient Certificates

The Patient starts by contacting his Certificate Authority, requesting the revocation of his compromised certificate and the creation of a new one. The Certificate Authority then revokes the certificate and updates his Certificate Revocation List to include the certificate the patient has requested to be revoked. Then the Certificate Authority issues a new certificate and forwards it to the patient securely.

The Patient can ask the VHCS to re-associate his Virtual Health Card with the new credentials. The VHCS will identify the patient's Virtual Health Card, since the new certificate has information that should enable the system to automatically re-associate. Information that can be used is the distinguished name and the identification number, the system extracting this information from the certificate and searching the Virtual Health Cards. If it finds a match, the system can update the tokens associated with that card. If not, human assistance will be needed. The first step is to set the revocation date to the previously activated tokens, then extract the information from the new one to associate it as a new token, and inform the patient that the operation has been successfully concluded.

After the patient requests his CA to revoke the credentials (step 2), any use of the compromised certificates should be easily denied, since services check the CRLs to see the certificate status. If the lists are not updated or the CA has become corrupt, the authentication will not fail in the first step, but will do so in validation by the VHCS, since the service can check the revokedData value on the tokens table of the service.

The revocation process for actor credentials is very similar to patient revocation. It only differs in the services used, as the actor uses the Broker service instead of the VHCS and

uses the actor certificate authority. When the need is to revoke a patient's internal tokens, it will be necessary to revoke and create a new pair of keys. Hence, the patient information needs to be deciphered using the old internal patient private key and cyphered using the new internal public key. This has to be done for all the information that is stored cyphered using the patient internal private key. The information is in the Accounter, Indexer and the Repository. The process is explained in Figure 3.21.

The VHCS starts by setting the revoke date of the internal keys. Then it generates a new pair of keys and stores them on the database. As the internal keys are used for cyphering the information stored in the Repository, Indexer and Accounter, the information has to be retrieved, deciphered with the old internal private key and cyphered again with the new internal public key.

The VHCS continues with the Indexer information, since the entire retrieval process needs access to the information in this component. The VHCS requests the Indexer for entries related to the patient, deciphers the fields that are cyphered using the old internal private key, and cyphers the same fields to the new internal public key. It then sends the data to the Indexer to update the changed fields on the Indexer database. After that comes the process of manipulating the bulks of data stored in the Repository. The Broker requests the repository credentials and retrieves the bulks. It deciphers each bulk, sends it to the VHCS which will decipher it using the old internal private key, cyphering it with the new internal public key. Then the bulk is sent back to the Broker, which cyphers it using the Broker key and then requests to store it using the same bulkid.

The final procedure is to change the information cyphered in the Accounter service. The VHCS asks the patient for accounting information, deciphers the fields that are cyphered using the old internal private key and cyphers the same fields to the new internal public key. Finally, the VHCS requests the Accounter to update the accounting information with the new cyphered fields.

The Broker uses a secret key to cypher the outer envelope before storing the bulk of data in the repository. If this key is compromised, it is possible that others can decipher the outer block of data. The information is still protected because it is also cyphered with the patient internal public key. Hence, the Broker needs a new key for his cyphering operations. It is necessary to re-cypher the information. The information must be retrieved from the repository, deciphered using the old key, cyphered with the new one and stored using the same bulkid. The process is explained in Figure 3.22.

Revocation of the patient internal private key means that all the patient's bulk of data in the Repository needs to be re-cyphered. This is a very time-consuming operation, although the system can simultaneously be active and functional while this operation is running. Hence, this operation is made in the background, using the old Broker key to access data bulks that were not already re-cyphered with the new key. The Repository Information stored in the VHCS has a flag to show each key still in use for a specific patient data. This flag states if the information in the Repository is using the old Broker key or the new one. In this process, the storage date of bulks in the Repository is checked. If it is stored after creating the new key, the new key is used, and if not, the old key is used. This will allow access to the information even if a specific patient does not have all bulks of data re-cyphered.

The Broker starts the procedure of revoking the internal key used for cyphering the outer envelope by generating a new key. It is inserted in the internal table, setting the previously active key to old and storing the creation date of the newly generated key. Then the Broker requests the VHCS to set a flag at all patients' repository information to show that the

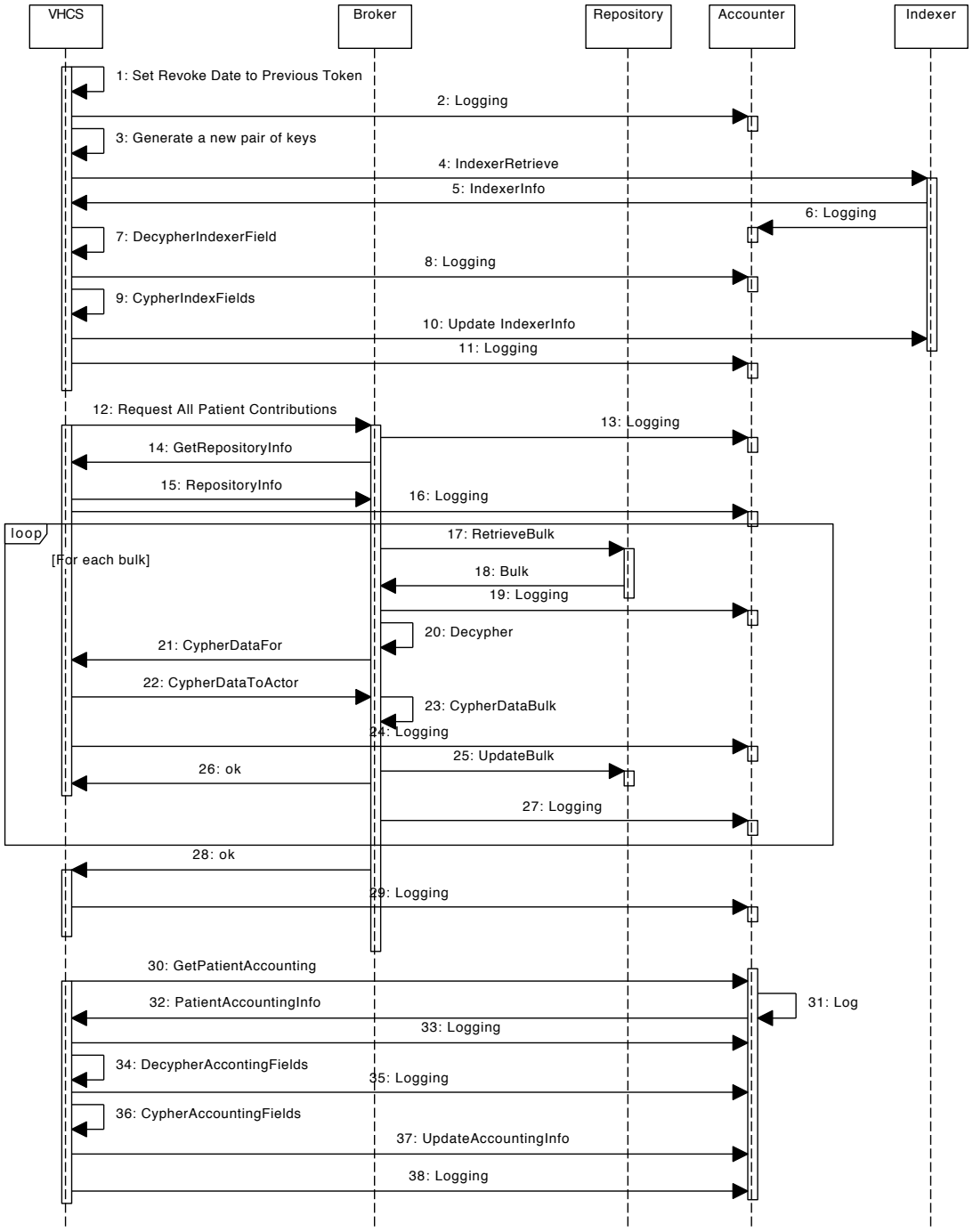

Figure 3.21: Revoking Patient Internal Keys

information about the patient is cyphered on the Broker using an old key.

The VHCS is asked by the Broker to re-cypher the outer envelope of all patient data bulks. Each bulk retrieved by the Broker is de-cyphered using the old key, cyphered with the new key and updated in the Repository. Finally, the VHCS is asked to set the flag to inform that this patient is using the new Broker key. This process enables access to the information

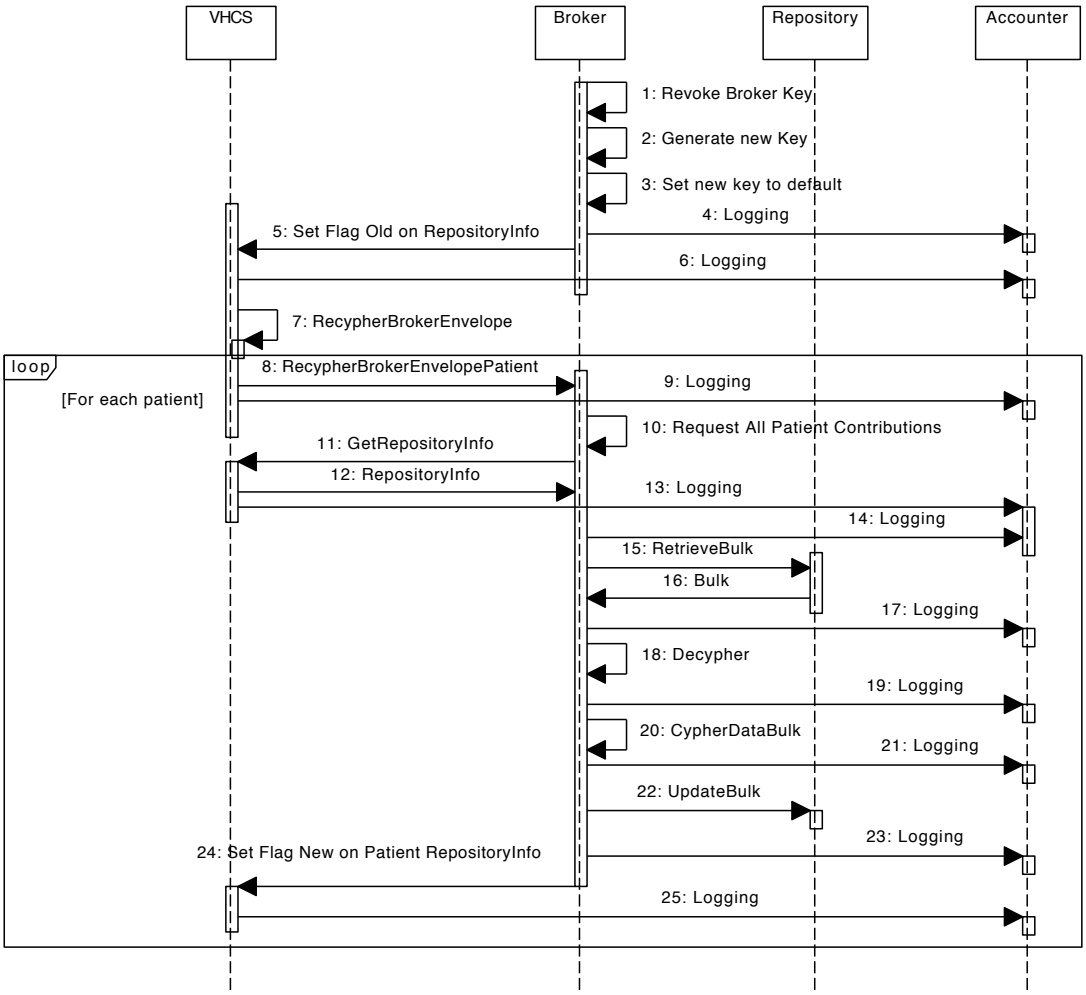

Figure 3.22: Generating a new Broker Internal Key

during updating of the data bulks in the Repository, since the store procedure is carried out with the new key, and during reading the flag is checked. If the flag is set to new, the new key is used, and if not, it checks the update date of the bulk stored in the repository. If the date is before the date of creating the new key, the old key is used. Otherwise, the new one is used.

After finishing for all patients, all accesses use the new key, ensuring that all information in the Repositories is protected against use of the compromised Broker internal key.

The credentials of the Repository can be also compromised. Thus, the information stored is protected by double cyphering of the bulks. The credentials should be changed. First, credentials are changed in the Repository and then updated in the VHCS. Since the Broker requests this information in every session, no more operations are needed.

Certificates can also be compromised. When this occurs, the PKI should append the credentials to the Certificate Revoking List (CRL). Hence, the service that has its credential revoked is no longer able to connect to other services, since every service checks the CRL when initiating the connection. The service should request a new certificate, and depending on the

service, may ask to revoke other compromised credentials. Once the Certificate Authority issues a new certificate, the service can connect again to all services.

3.3.7 Digital Signature Validation

The integrity of the produced and stored data is achieved using digital signature mechanisms. Concerning data in the repository, the actor producing the information signs it using his certificate before the store procedure starts. So it is possible to verify if the information deposited was created by a trusted actor.

The information stored in the VHCS needs integrity validation and non-repudiation by patients. Therefore, the VHCS database stores hash results of the signatures of all the concatenation values stored. This enables verification of the information in the demographic, groups and policy tables. The VHCS has a procedure for checking the signature of some data. Using Algorithm 2, the method receives the data to check, the hash stored and the time-stamp of when it was stored.

Providers also sign their information when creating their accounts with the Broker. They sign the information related to their contact information, numbers and description of services they provide.

The information created by the actors is also ciphered using their certificates. Here, the serialized data bulk in XML and the metadata are signed before being cyphered with the patient internal public key. The bulk is stored in the Repository and the metadata data on the Indexer. Both are validated on retrieval. The information on the Indexer, when it is retrieved to the VHCS for making a query, has its signature checked. The signature of the information stored in the Repository is also checked during the retrieval process. Thus, during reassembly, each bulk signature is verified. After this process the resultant bulk is signed using the VHCS certificate.

The Accounter also has signed information, with all fields having a complementary field for storing the result of the signature function which uses the certificate of the service that generated the information. The accounting information is also cyphered for the internal patient public key.

One challenge is how to store signed data over a long time, since certificates will expire or will need to be revoked after the signing process takes place. The approach followed was to store all the certificates. The VHCS stores all the patient certificates, the Broker all the actors' certificates and PKI services those related to architecture services certificates. These lists are associated with the date of activation and revocation. Hence, when validating a signature, the method receives the data, the stored hash of the signature and the date and time of the signature. The date and time will enable a search for the public key used by the actor at the time of signing, enabling the generation of the hash for the data stored and comparison with the stored one. So services can check the integrity and the creator of the information even after the certificate has expired or been revoked.

3.3.8 New Services Integration

One of the goals of the proposed architecture is to enable collaboration between all the actors involved in healthcare or healthcare-related activities. Hence, the architecture allows the creation of new services to manipulate or create information in the patient repository, for wider collaboration. The new services are external to the architecture, using the Broker

service as an interface for communicating with the full architecture. The actors contemplated during the architecture modeling can be human actors interacting with the low level methods provided by the web services of the Broker and the VHCS, or automated software clients interacting with those services to create a richer experience for users or bring functionality to the architecture. The new services can be service providers that give the patient access to more information or they can be automated processes, applications or agents helping the patient to manage and exploit his information.

External services can be divided in two main categories: consumers and producers. Both are seen as actors in the architecture and so services just need to register on the Broker as a service, using a certificate signed by a trusted Certificate Authority. Then, when authorized earlier by the patient, services can store or retrieve information for their purpose, using the low level methods available on the Broker.

One important aspect is related to services that create information. The service has to check if the archetype needed for the type of information that they are creating exists in the OpenEHR Archetypes Repository. If it does not exist, the user or service should store it in the Repository prior to any data operation.

Another important aspect concerning retrieval operations using external services, is that information privacy should be ensured by the external service. Since information is cyphered before leaving the VHCS to the requesting actor public-key, i.e., to the external service key, the service will be able to manipulate the patient information in plain text.

The architecture also enables integration of new services in the core of the architecture. These services require a prior certificate issued by the Certificate Authority of the architecture. They are normally components inside the VHCS allowing data manipulation in plain text or extending the requests an actor can make to the architecture. The components should respect the BrokerInterface of the VHCS, and after deploying the service inside the container, the new methods available to the Broker are registered on this interface.

Future development of services related to patients' management of the virtual health card are expected. Those services also need to request a certificate and are deployed inside the VHCS, each one inside the corresponding container: Repository Management, Patient Credentials Management, and Policy Management.

3.3.9 Availability

This architecture deals with critical information and should address high availability and fault tolerance. Connectivity is a strong point of failure that should be addressed by actors needing access to services. It they work in emergency care, backup connections should be in place.

The proposed model, a Service Oriented Architecture, is based on self-contained components, aggregated in service containers which are executed without the need to store results for future invocations. Another important design decision was the separation of the data layer from the service layer. This aspect permits different policies related to data backend replication and service layer replication.

Repository services should be provided externally by other players in the market. The patient is free to choose what provider to use. The repository provider chosen by the patient is responsible for high availability and fault tolerance. To enable a solution with high availability and fault tolerance, service and database replication are considered with the aim of achieving more resilience and performance solution.

Database replication works not only in enabling the architecture's performance and fault tolerance, but also as an alternative to most backup solutions. If a database instance becomes corrupt, the services that make use of that database are forwarded to another replica. Hence, a new replica can be created, using another in perfect condition. Most database management systems support solutions for snap shooting and roll-back that help databases to recover from errors, enabling the database to be restored to the last stable state.

The prototype is deployed using J2EE technology on an application server. Most of these services support clustering. Clustering is an easy method of allowing the architecture to scale, increasing the performance and availability of the service layer.

Considering the back-end of services, the cluster solution can also be used to increase the availability and performance of database level operations, therefore increasing the overall performance and availability of the architecture.

Replication of the services layer is not a critical procedure since this can be done without the need for synchronism between replicas and propagation of replica states. However, the challenge arises with the need to replicate the databases, as each operation that creates or updates information on a database has to be propagated to other replicas.

Some database management systems support data replication. The implemented architecture is based on a solution that mediates access to the database through a specific container. Therefore, low-level replication control at the database level can bring some data inconsistency between the services that use the various replicas. Since the application services make use of caching mechanisms, at the database link and container levels, these caches have to be invalidated whenever the database itself is changed by propagation of changes in one replica. Therefore, the solution of replication should be implemented at the database mediation access level of the application server. With this approach, on data creation or updating, a trigger can be launched to propagate the changes to all replicas and invalidate the caches of the services that make use of the database replicas that were updated.

3.4 Summary

This chapter presents a solution that copes with mobility by addressing the challenges of collaboration.

First the business logic associated with a platform that copes with the open and secure collaboration between all the actors that the patient desires was defined. The approach was based on the combination of the strengths of EHR and PHR oriented for clinical integrity, collaboration and patient control.

Secondly an architecture was modeled to implement the business logic required, also addressing the security challenges. This architecture was modeled to enable easy access to new providers and patients, with freedom of choice of repository for storing the patient record and based on open standards for enable the development of new services that can use the architecture.

The following chapter analyzes how the requirements where addressed, it also delineates a methodology for security analysis and makes an assessment of the security of the proposed solution. The security validation of the prosed architecture is critical to verify that required security properties are respected and to provide feedback to correct them before going further with the development. Enabling detection and correction of problems in early stages minimizes future detection of structural security problems. Another result is the description of a

common methodology and mitigation techniques that can be used for similar architectures.

Chapter 4

Discussion

4.1 Coping with Requirements

The solution proposed in the previous chapter was designed to cope with requirements of mobility, collaboration, openness and security. Mobility requirements were addressed by enabling the collaboration of all actors. For effective collaboration, it should be available to all actors, it should be open and should use the most common standards in information technology. The proposed solution should also have the flexibility to support future demands.

Collaboration is achieved by enabling all involved players to access patient repository. Since they are worldwide, the architecture supports different Certificate Authorities. Initially, it will support the most commonly used ones, preferably CAs in countries with Public Key Infrastructure for issuing electronic identification cards (e-id) with certificates. Patients can use their e-id for auto-enrolment and for authentication purposes. After auto-enrolment, providers need manual validation of their information. This approach ensures trust between providers.

Providers contribute to a patient central repository using OpenEHR contributions corresponding to the information created on the patient. Using the OpenEHR archetypes enables semantic validation of the data. Data signing brings clinical trust to the information stored in the repository. Providers can keep using their systems to store their data, only needing to create a service that uploads a signed resume, in this case using the OpenEHR format.

The contribution is not restricted to healthcare professionals. Providers and users considered by the patient can boost his electronic health record by creating and using information. Those actors can extend the type of information by proposing new archetypes. It is possible to know who created the information and to check its integrity, since all contributions are signed, thus enabling clinical trust of the stored information. All contributions will be stored in a repository whose provider the patient can freely choose. The proposed solution has complementary services to ensure security regardless of the repository as our approach protects against disclosure of information in the repository, even from the repository system administrators. The repository provider will be responsible for replication and backup, for high availability and performance of access to this service.

The success of this approach is based on the active collaboration of all actors. The limitation to collaboration is only caused by the policy managed by the patient. To lower barriers to the development of new applications by new players or even by providers, openness was a requirement.

The openness of the solution was achieved by adopting widely used standards in information technology and making them compatible with the medical area. Use of the web service with publicly available interfaces enables easy collaboration to create new services and applications which can interact with the services of the proposed architecture. Use of the XML standard makes it easier for developers to manipulate the information, as they use the normal tools for data processing. Combining with the OpenEHR archetype, semantic constraints can be enforced in the stored data. Storing the data in an XML format allows protection of the data structure and the data simultaneously. The OpenEHR framework is open source, enabling collaboration in its development, customization and it is vendor-free.

Another important aspect of systems that deal with long store information is the capability of future evolution. This support is based on the OpenEHR archetype approach that enables the coexistence of different versions. Also, if a new type of information needs to be managed, the actors involved can collaboratively create an archetype to be used for that information and share it in the OpenEHR archetype repository. As the solution deals with transparent bulks of data, the core services do not need any changes to enable and use of new archetypes or updates.

To verify the applicability of the proposed solutions, a brief explanation of some uses follows. Some common operations were detailed previously in Section 3.3. These operations support some of the use cases that will now be demonstrated.

Access to the architecture is based on the creation of an account. The patient has first to create his account with the VHCS and providers need to create an account with the Broker, as illustrated in Figure 4.1. A patient should first choose a service provider to work as a back-end of his repository.

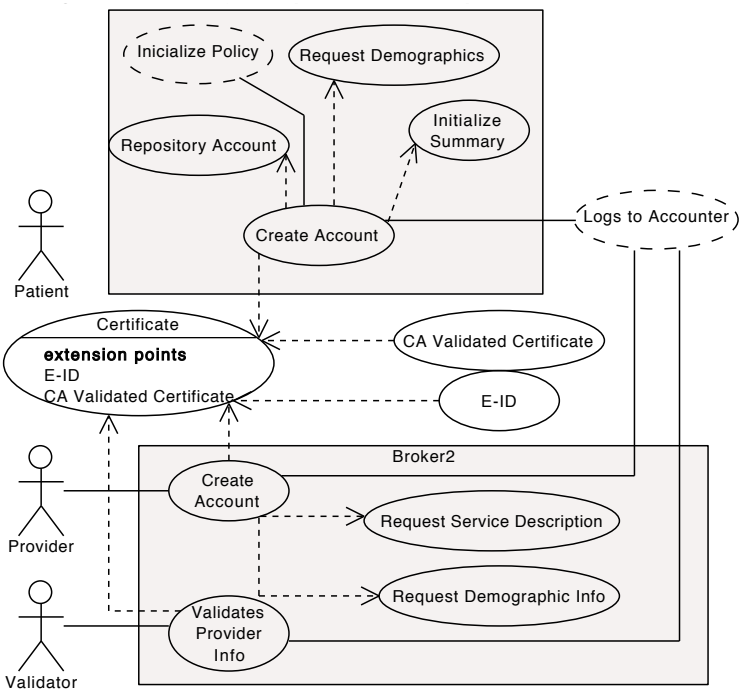

Figure 4.1: Creating accounts

After having an account in the repository, the patient should created an account in the

VHCS using his e-ID. After creating the account, he will be requested to store information related to demographics and information to access the patient repository service. Then the patient allows services to manage information on his behalf, based on the policies he defines.

Providers should create their account with the Broker, using their certificate in an auto-enrolment process. They give their information and a statement of the services provided. This information is signed using the certificate.

The working procedure is based on the patient's desire to give a specific provider access to his record. The patient searches in the Broker for providers and then, in the VHCS, configures a policy for the chosen provider. When this process ends, the provider can start to collaborate. The process can also be triggered by a provider requesting access to a specific patient. At this stage, the patient receives the request, and if he wants to grant access he creates a policy for the requester. This process is illustrated in Figure 4.2.

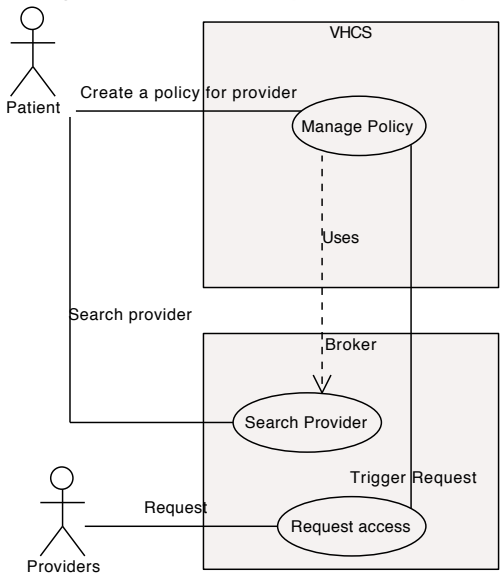

Figure 4.2: Managing Policy

The provider needs to verify if there are already existing archetypes for the information that he manipulates in the repository. If the archetype exists, he can use it. If not, he has to create one, preferably in a collaborative process with other providers from the same area to propose a consensual archetype that will benefit all. This approach will enable new providers to collaborate in innovative ways.

This will allow non-medical providers to store information which they manipulate with clinical information. A patient who is being accompanied by a doctor, alternative medicine provider, physiotherapist or sport monitor could benefit from closer collaboration between all the providers. In Figure 4.3, some common operations are explained. Each provider should verify that there is an archetype for managing the information.

This enables other collaborators providing patient care to read the information or create new information respecting the semantic constraints. The patient can also collaborate by making annotations using a correspondent archetype. New ways of monitoring will also appear. Players who create these new ways can create a consensual archetype or use one already defined to upload the information directly to the patient repository, using the pre-

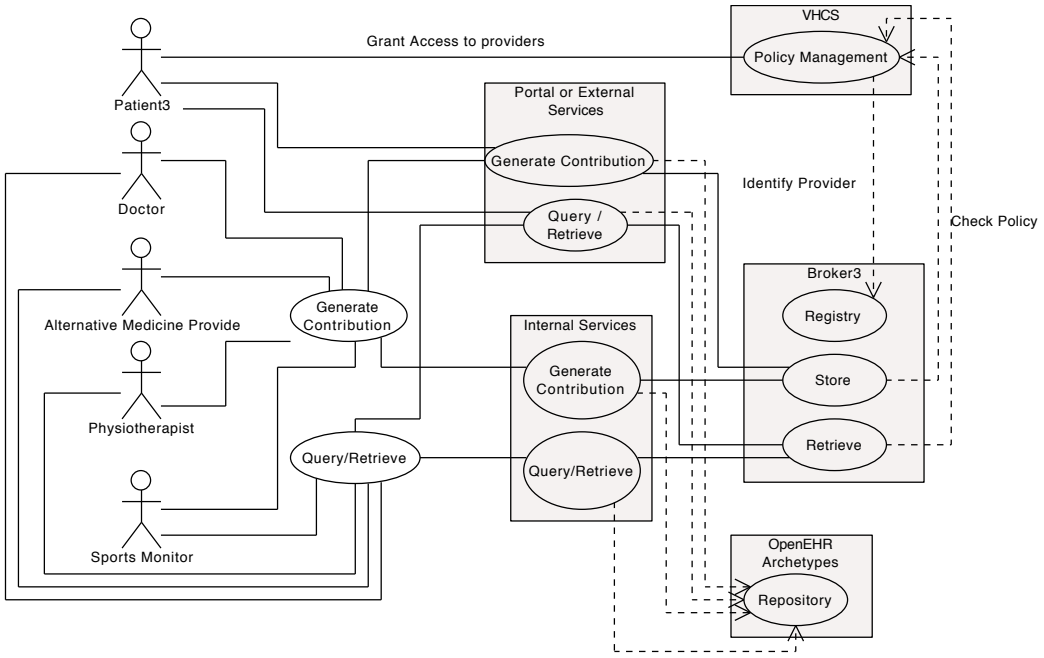

Figure 4.3: Actors' collaboration

established interface and protocol.

In the above depicted scenario, the patient needs to configure a policy to enable actors to collaborate. The patient searches for each provider and creates the policy for each one. Specifically, he could decide to create a policy based on the roles and sensitivity of information, or based on a group. The patient's choice depends on the granularity level of the policy that he wants to enforce.

When the patient goes to a provider, and the provider has an internal system for managing records, the provider could create information in his system and later upload a summary using the correct archetype for the architecture. If not, he could use a portal or other external service to create the information or generate contributions in the form the architecture stipulates. For search and retrieval, external services or a portal could give access to the requester to receive information. Also, if they respect the interface they could query the Broker for the internal system to retrieve information for some patient.

This analysis verifies that the proposed solution copes at a functional level, satisfying the requirements that emerged during the problem analysis. Considering that security is very important and the security model is an important part of this dissertation, a security analysis of the proposed solution follows.

4.2 Security Analysis Methodologies

The security of a solution depends on all its components and dependencies, both physical and logical. Hence, the security of applications, services, systems and networks concerns and depends on all actors involved in its creation, maintenance and use. There are three main vectors of data security: privacy, integrity and availability. Those can only be achieved if all the application layers address those challenges implementing suitable controls.

To address these challenges, Jones and Rasto [108] recommend that security should be incorporated in the Software Development Life Cycle, making security transversal to all the development phases. In the design phase, the security context and security requirements should be decided. A risk assessment contemplates asset identification and the creation of a threat model should be considered. Risk mitigation procedures should be carried out on the threat model and a security review made. During the development phase, the developer should be aware of security risks, and should use secure coding patterns and approaches. In the testing phase, unit testing can be used for code reviewing, integration and quality assurance testing, penetration testing, and optionally certification of the system. It is also important to consider the operations and maintenance phase's specific procedures related to security, such as security training, management of cryptographic material, management of privileges, monitoring and others. The disposal phase should consider operations connected to the movement, sanitization, disposal and archiving of software, data and hardware.

Threat modeling can be seen as a core step to achieve a secure solution [108], as it is in this phase that an application model is created and the nodes that can become a target of security attacks are identified. The procedure comprises the following phases:

- Each target node is analyzed for attacks
- Construction of an attack tree describing vulnerable system areas to specific threats
- Risk mitigation phase – Analysis each identified threat to answer the following questions:
  - How an attack arises
  - How the vulnerability can be exploited
  - What are the suitable controls to mitigate the threat

Different methods and tools exist for threat modeling and analysis, such as misuse cases and attack trees [109]. The misuse cases are use cases that represent misuse behavior (attack) from attacker to systems, enabling to reason over the vulnerabilities that are exploited on each attack and how each vulnerability can be mitigated. The attack trees enable to create a tree that shows how an attack is performed, being the root the attack objective and the arrangement of the leafs (each one represent sub-attacks needed to achieve the propose) represents possible vectors of attack.

Historical data related with attacks, vulnerabilities and weakness, configuration errors enable security experts to understand how systems and its components can be attacked. This information is available in security databases such as National Vulnerability Database (NVD)[1], Common Weakness Enumeration (CWE)[2], Common Attack Pattern Enumeration and Classification (CAPEC)[3], and Common Configuration Enumeration (CCE)[4] [12]. The attack patterns give information about how an attack works. It has a unique pattern name and classification, the attack prerequisites, description, analogous vulnerabilities (uses common Vulnerabilities and Exposures (CVE) number[5]) or weaknesses (uses CWE). It also describes the method of attack, attack motivation-consequence, attacker skill or knowledge required, resources required, solutions and mitigations, context description and further references [110].

[1] `http://nvd.nist.gov/`
[2] `http://cwe.mitre.org/`
[3] `http://capec.mitre.org/`
[4] `http://cce.mitre.org/`
[5] `http://cve.mitre.org/`

When analyzing the possible attacks to a node, the STRIDE approach can be used. The STRIDE approach represents Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of privilege. Using this systematic approach, some security properties can be verified [11]. The process is based on the analysis of how each component behaves when under each attack classes. For instance, the security expert can search for spoofing attacks to architecture functionalities, using historical databases or personal expertise and verify if that attacks will succeeded. If it does the related security property is compromised. The relations between threats and security properties are illustrated in Table 4.1. These relations enable us to understand which threats the systems need to mitigate to support the security properties.

Table 4.1: Threats and Security Properties [11]

Threat	Security Property
Spoofing	Authentication
Tampering	Integrity
Repudiation	Non-repudiation
Information Disclosure	Confidentiality
Denial of Service	Availability
Elevation of privilege	Authorization

The security assessment of an architecture can be made using different approaches. A good survey [111], divided the approaches in the following areas: capture of the security architecture, techniques for discovering attacks and techniques for comparing and assessing security architectures.

One of the challenges of ensuring security in all phases of software development is the knowledge gap between security experts and developers. To narrow that gap, with the purpose of reducing the occurrence of security vulnerabilities, the SHIELDS project[6] arose. The project brought new and innovative security inspection and vulnerability detection tools, integrating security aspects and activities into software methodologies, advancing security modeling technologies [12]. Collaboration was addressed by the creation of a central repository (called the SVRS) that enabled the sharing of security models. Those models can be used in development tools to automatically detect and eliminate possible vulnerabilities [112]. The models should be developed by security experts. SHIELDS has different models with diverse purposes summarized in Table 4.2.

SHIELDS proposes six different activities to improve security during software development:

- Security Goal and Vulnerability Class Identification
- Goal Driven Security Inspections
- Vulnerability Driven Security Inspections
- Selecting Mitigation Strategies

[6] `http://www.shields-project.eu/`

Table 4.2: Overview of Models [12]

Model	Purpose	Relation to other models
Misuse case	Get an overview of typical threats towards functionality commonly found in software systems, and common countermeasures.	Provide input for finding relevant VCGs, SGITs and VIDs.
Attack tree	Get an overview of how an attacker can achieve a specific attack goal, in order to protect a system from such attacks.	Provide input for finding relevant VCGs, SGITs
Vulnerability Cause Graph (VCG)	Improve understanding of software vulnerabilities by identifying a vulnerability's causes and their relationships.	Will provide Causes and paths leading to a vulnerability that will allow defining VDC (see below).
Security Activity Graph (SAG)	Identify software development activities that can prevent vulnerabilities by addressing their causes.	A SAG is typically associated with a cause or a security activity.
Vulnerability Detection Condition (VDC)	Formally describe system or application behavior that allows proving that a cause is present in the implementation and execution traces. This information is then typically used by testing tools to perform automated vulnerability detection.	Can be linked a VCG; several VDCs can be derived from a VCG.
Security Goal Indicator Tree (SGIT)	Describe indicators that can be examined to find if a security goal has been correctly implemented. The structured set of indicators can then be used to guide inspections.	Can refer to indicator specialization trees, and are related to security goals.
Security indicator specialization tree	Give more details on an indicator that can be used for inspections, e.g. how to check for this indicator in different document types and on different platforms.	Connected to SGITs.
Guided security inspection checklist	Provide an easy to use guide for how to inspect whether a security goal has been correctly implemented.	Created based on SGITs and indicator specialization trees
Vulnerability Inspection Diagram (VID)	Guide inspections for a specific class of vulnerabilities.	Related to vulnerabilities.
Security inspection scenarios	Give concrete guidance as to how to perform the actions described in a VID in order to inspect for the vulnerability.	Created based on VID.

- Vulnerability Cause Presence
- Security education of vulnerability Causes

As previously mentioned, security should be integrated in all phases of software development cycle. During the design phase an analysis should be made before starting the development phase. The security review made in this phase allows the detection of design choices that could block the implementation of controls to achieve the required security properties. Incorrect design choices can compromise future development, making impossible to implement required security properties as they can be incompatible. This verification is more complex when the systems use services composition and the security properties need to be enforced among different provides. The security analysis made to our proposed architecture is based on the creation of a high-level threat model to enable to verify how it copes with common threats to the security properties. Considering that the Security Goal and Vulnerability Class Identification phase of SHIELDS enables the creation of a threat model, this activity will be explained in detail. For threat modeling, the forms used in SHIELDS are misuse cases and attack trees. To create a high-level threat model, the proposed workflow is illustrated in Figure 4.4 [12].

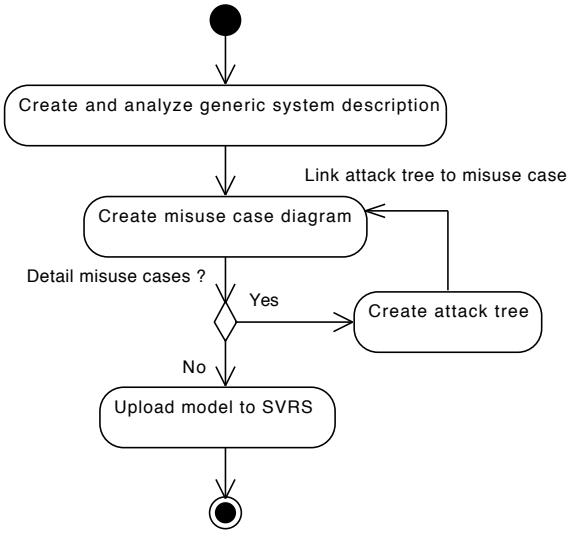

Figure 4.4: Example of a high-level threat modelling process

The phase of creating and analyzing generic system description can be performed with a use case model of the software, or by identifying all the verbs in the system description, following the identification of nouns to understand what actors and assets are. Finally, the functionality of the system should also be identified.

The next phase is the creation of misuse cases. SHIELDS proposes two alternatives, firstly the use of pre-existent misuse case models in the SVRS and the other is modeling from scratch. In the first approach, using the centralized repository, a search is made to select existent models that are analogous to the system being developed. Then the models are imported and updated to cope with the functionality of the new system. The inclusion of new threats is made by brainstorming or by updating the model with misuse case stubs, as the second alternative. In the misuse case, countermeasures should be proposed to mitigate

threats. If it is useful to explain in detail how an attack can be made, an attack tree can be associated with the misuse core elements.

The second approach to creating misuse cases is to use categorized threats and misuse case stubs as a source of creating new misuse cases. First, a use case for the software should be created with the functionality. Then for each use case, brainstorming is used to select relevant threats that can influence the functional properties of the system and to find security activities that can mitigate those threats. To search for possible threats, a list of known attacks and threat categories can also be used, such as the STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of privilege), threat taxonomies, textbooks and own security expertise. Hence, each misuse case is connected to the use case using a threatening relationship. If the use cases have potential vulnerabilities, the misuse case is connected to vulnerabilities using the exploit relationship. Then security activities are associated with the misuse cases using the mitigating relationship to identify how the threats can be mitigated. This process may be repeated several times, as security functionality appended to mitigate can provoke new threats and attacks. Optionally, a textual misuse case description can be created elucidating in detail the threats with description, motivation and mitigation activities shown in the misuse case [12].

The other phase of high-level threat modeling is the creation of attack-trees to explain how an attack can be carried out. Attack trees enable developers to understand how attackers work and how common exploits can work against the system. They enable the developer to have a better understanding of attacks and attack patterns without the need for thorough knowledge of security, enabling him to be better prepared to define mitigation strategies. The first step, in creating attack trees, is to decide the attack goal that is going to be modeled, and include it as the root node. Then the diverse attack paths to achieve that goal are modeled as child notes of the tree. Brainstorming is used with different sources of information, such as information from real life incidents, security knowledge, and databases such as the CAPEC or others referred to earlier. The paths can be combined with alternative paths, or if some of the child paths have to be achieved to reach the parent, the child nodes are aggregated using the AND gate. This should start with the topmost attack goal node. When the tree is complete, an analysis should be made to verify what security activities can mitigate the attack branches. Those activities should be appended to the misuse case diagrams as security use cases [12].

All the models produced, in this case the misuse case and the attack trees, should be shared for feedback in order to improve the models. Good models will allow the creation of a more secure system, based on model analysis and checking [12]. The SHIELDS project enables collaboration through the centralized repository SVRS.

Different tools were created to enable the SHIELDS approach. The SeaMonster [113] is an open-source tool compatible with SHIELDS approaches and their SVRS repository. It permits the creation of misuse cases and attack trees, enabling security to be modeled, with different linked viewpoints of vulnerabilities, facilitating the revelation of causes, threats and countermeasures within the same tool.

4.3 Security Analysis

The security analysis of the proposed solution is made resorting to a threat model, employing misuse cases and attack trees.

The actions considered for creating the threat model were high-level. For the Patient, the following was considered: Manage Repositories, Manage VHC, Manage Access Policy. For the User/Service, Registration was considered. In both type of actors we have considered the Store Data, Request Data and Authentication services. The functions provided by complementary services of the Certificate Authority (CA), OpenEHR archetypes will not be represented in the misuse cases, as they implement common functions. The CA is self-managed and enables common services to the architecture components. The OpenEHR repository stores signed archetypes that each components or actors can request to retrieve without authentication. The storing of new archetypes is done manually.

Using the SRVS models, with databases with past security data and own security expertise, we construct the high-level threat model for security validation of the proposed model. The misuse case models reflect a subset of threats from the STRIDE analysis.

Figure 4.5 illustrates a subset of a misuse case model for contemplated threats. In this model, for better readability, the *threaten* legend is not present. Here, the most common misuse cases (ellipses with a black background) are represented. This model is an overview of the considered threats coming from an analysis of the information in the SRVS repository. The attacks contemplated in the Create/Use information use case are:

- Modify Database
- Replace Content
- Make Service Unavailable
- Create False Message
- Spoof server
- Spoof user service
- Replay another users' data packages
- Modify Data in Transit
- Eavesdrop
- Access to private/secret data

For the registry were considered the following misuse cases:

- Replace Content
- Make Service Unavailable
- Create False Message
- Spoof server
- Spoof user service
- Replay another users' data packages
- Modify Data in Transit

- Eavesdrop
- Access to private/secret data
- Create and use fake user account

Analogous, to the manage use case, were contemplated the following misuse cases:

- Replace Content
- Make Service Unavailable
- Create False Message
- Spoof server
- Spoof user service
- Replay another users's data packages
- Modify Data in Transit
- Eavesdrop
- Access to private/secret data
- Create and use fake user account

The authentication action is requested by other use cases and was considered to be threatened by the following misuse cases:

- Replace Content
- Make Service Unavailable
- Create False Message
- Spoof server
- Spoof user service
- Replay another users's data packages
- Modify Data in Transit
- Eavesdrop
- Access to private/secret data
- Create and use fake user account
- Steal Digital Identity
- Steal Session
- Get access to administrative operations

- Bypass access control
- Take control over user terminal
- Steal user terminal

A deeper analysis follows, by creating a more comprehensive misuse case for each of the STRIDE vectors. Figure 4.6 considers the subset of misuse cases related to spoofing. This shows the misuse cases linked to the vulnerabilities (dotted line ellipses) that are exploitable and linked to security activities (grey ellipses) that can mitigate those misuse cases. The base definitions and mitigations came from the SVRS repository of the SHIELDS project.

In this context, spoofing is a method of attack which tries to get illegitimate advantage by masquerading as another person or software component. Considering the proposed architecture, the misuse case for spoofing is resumed in Table 4.3.

Table 4.3: Misuse case diagram: Spoofing

Threat	**Description**	**Motivation**	**Mitigation**
Steal Session	Sessions are utilized to be able to keep state information in a web application, and by stealing the session of an authenticated user it will be possible to get unauthorized access to resources.	An attacker tries to steal a session already established by a trusted user, or a sessions established between each architecture services and components, with the propose of use the services as the previous authenticated user.	Use sessions expiration and protect credentials. Make sure sessions have limited lifetime to avoid that attackers can reuse them. Protect e.g. passwords and keys when stored and when transmitted over a network.
Steal Digital Identity	An attacker steals user credentials, for enabling him to authenticate using a trust user account.	By stealing a user's digital identity it will be possible to spoof a user identity and get unauthorized access to system resources.	Protect e.g. passwords and keys when stored and when transmitted over a network. Do not rely on password based authentication, but rather use two-factor authentication by e.g. also using digital certificates or one time passwords. The proposed solution uses two-factor authentication, using certificates protected by password, some of them securely stored outside the systems on a crypto-card as on Electronic Citizen Cards.

Continues on next page

Table 4.3 – *Continued from previous page*

Threat	**Description**	**Motivation**	**Mitigation**
Spoof the User Service	Attacker can offer a false service similar to the original one.	The attacker tries to deceive users and services to interact with fake service. With the propose of gaining access to privileged information, credentials or send false information for gaining illegitimate advantage.	Verify the server's identity by using e.g. signed digital certificates. The proposed solution, uses an internal Certificate Authority for controlling the issued certificates for the architecture services. Services use signed certificates to authenticate themselves, services use mutual authentication using the issued certificates.
Spoof Server	Attacker can set up e.g. a fake web server that replicates the graphical user interface of a legitimate server.	The attacker tries to deceive users and services to interact with fake server. With the propose of gaining access to privileged information, credentials or send false information for gaining illegitimate advantage	Verify the server's identity by using e.g. signed digital certificates. The proposed solution makes heavy use of mutual authentication using the issued certificates.
Take control over user terminal	Attacker gets illegitimate access to a user's terminal	The attackers tries to gain access to users terminal, to have access to information to enable him to access to user services.	Control who gets access to resources, and make sure access control mechanisms cannot be circumvented. The user's terminals were not considered in the architecture.

Continues on next page

Table 4.3 – *Continued from previous page*

Threat	**Description**	**Motivation**	**Mitigation**
Steal User Terminal	Physically steal a user's terminal	Tries to gain access to privilege information, that will enable him to impersonate the user that owns the terminal.	Control who gets access to resources, and make sure access control mechanisms cannot be circumvented. Encrypt the hard disc content of a terminal. As earlier stated the terminal security was not considered, but the design protects the credentials if the users authenticate recurring to Electronic Citizen Cards, or external dongles, with cryptographic protection.

In the proposed solution, the credentials stored in the architecture make use of cryptographic mechanisms. Users' certificate authentication is also used, making it very difficult to steal a digital identity via the server side, since the server/service has the public-key and it is computationally infeasible to compute the private part from the public part. The VHCS service has two components, Patient Credentials and Repository Management, which store some credentials. If those components and service are compromised by administration privileges, those credentials could be accessed.

Service or server spoofing is mitigated using mutual authentication of client application and services using certificates. As such, every service and components of the architecture perform mutual authentication before any communication negotiation between them. An internal certificate authority issues these certificates. Also, users' authentication in the VHCS and Broker is assured through certificates issued by trusted Certificate Authorities.

Another aspect contemplated is related to tampering. Tampering is interference aiming to cause damage or make unauthorized alterations. The tampering misuse case is shown in Figure 4.7 and resumed in Table 4.4.

Table 4.4: Misuse case diagram: Tampering

Threat	Description	Motivation	Mitigation
Modify database	The attacker modifies the underline databases that are used to store information to the applications and services.	Many applications and services rely on databases for data persistence. This approach can be more effective than attacking the front-end of the services, as usually implement less security mechanisms and are at mercy of an insider. The attacker tries to gain access to the database, and modified it, in other to change the business logic of the application or service.	Log all activities. By logging all activities, prohibited activity can be detected. Perform strict database access checks. Do not let anyone read, add, modify or delete data to the database without proper access control. The proposed model makes use of digital signing and cyphering of data present on the underline databases to enabling detection of abnormal modifications of the data.
Replace content	An attacker replaces the original content with malfunctioning or malicious content.	The attacker has two types of motivations. First, he want's to inject malicious code, for trying to execute commands on client side, or on service side when parsing the contents. Second, is to modify the information, that change the normal function of the services. E.g. change the repository information, forcing the Broker to store the new patient data bulks in other repository.	Perform strict database access checks. Use data hashing and signing on all content.By hashing and signing all content the users will be able to verify its origin.
Create false messages	An attacker inserts false messages in an electronic conversation.	The attacker creates messages, for educing other services to execute requested actions as it were requested by a valid service, application or actor.	Use data hashing and signing on all content, and also certificates to mutual authenticate the services and application.

Continues on next page

Table 4.4 – *Continued from previous page*

Threat	Description	Motivation	Mitigation
Modify data in transit	The attacker, tries to intercept and modify network traffic between systems. E.g. using web proxies or making an attack of the type of Man In the Middle.	The purpose of the attacker is to change the information that a system is sending to another, to work for the attacker proposes. E.g. if the message is a command to grant access to user X to a record, the attacker can try to change the message to grant access to the user attacker.	Sign data in transit. Signing data in transit makes it impossible for an attacker to modify the data without detection.

One form of attack is on the VHCS, to gain access to modify the supporting database of the Patient's Policy, so as to manage the access policy for the patient's records. The patient credentials database could also be changed, to corrupt patient access to the platform, or the public token changed to enable the attacker to use one for which he holds the private part, giving him access as the patient. The attacker can also try to modify the contents of the Repository Credential database in the VHCS, forcing the Broker to initiate a new patient repository to store contributions.

Modifying databases that support the Broker's normal operation could enable the attacker to act as a provider by changing the public token, allowing him to use one for which he holds the private part. He could also change the type of services provided, the provider's information and more.

The attacker can also target other services, such as the Accounter, Repository or Indexer. On the Accounter, the attacker can try to change the logged information stored by other services to camouflage attacks on the architecture services. The attacker can modify the data in the Repository to create false data for the patient, and can also try to change information on the Indexer, to disturb the normal process of information searching over the patient repository.

These types of attacks are mitigated by predominant use of cryptography. The information on databases and in repositories is usually signed using the token used by the actor and cyphered to the patient internal public-key, for patient-related information. Therefore, the change of information on databases, change of content, creation of false messages or change of the information in transit can be detected and refused by the architecture services. Hence, access to the services of Accounter, Repository, Indexer, Certificate Authority, changing databases on the VHCS or on the Broker (Internal) is controlled by mutual authentication, making it harder for the attacker to access them directly and launch an attack. The attacker has to gain access to a valid certificate to allow communication with those services.

The repudiation misuse case diagram is illustrated in Figure 4.8 and resumed in Table 4.5. Repudiation is the concept in which a user can dispute a transaction or a request to a system. Normally, this is associated with insufficient logging and auditing information, or the ability to verify the identity of the requesting user. This analysis contemplates the attempt to act as another user, by creating a false account or replaying data packets or other users.

Table 4.5: Misuse case diagram: Repudiation

Threat	Description	Motivation	Mitigation
Create and use fake user account	If an attacker creates a fake user account it can be difficult to trace an abuse back to the legal person or entity.	The attacker tries to deceive the system, by creating an account using fake information, making very hard to trace to the legal person. This will push the attacker act on the system has untraceable. Or traceable to other person or entity.	Request email confirmation. For systems that need a strong binding, they can require identification document. Require that users prove their identity by showing an identification document.
Replay another users's data packets	Repudiation can be challenged by users replaying previous packets from another user to get some information or perform some actions on a system.	The attacker tries to send previous packets for making the same request that an authorized user had done in the past.	Use session expiration. Validate freshness of each packet. Perform checks on the freshness of packets in order to detect packet replay.

Some approaches consider using an email address to validate user information. The user needs to have access to that email account to validate the email address. Usually it is an email that contains a URL or a token needed to confirm the action. This approach is used for some services that do not request a strong bond between the legal person or entity and the account being created, as false e-mail accounts can be easily created or even corrupted. This approach may be weak for the creation of e-banking accounts and other critical services. As countries are adopting the use of Electronic Citizen Cards, the proposed solution uses the token on those cards to create the accounts. The user is requested to use the certificate inside those cards and asked for a PIN or password for identification purposes in the self-enrolment process.

Repudiation is also minimized by signing all the requests and data with the token used by the requesting user. The services use mutual authentication, which makes it more difficult for an attacker to replay previous data packets, such as the capture of packets in communication between services. Electronic Citizens' Cards are also used for reliable authentication of the person creating the accounts in the systems, enabling the legal person or entity to be traced.

Another important aspect to be analyzed is information disclosure. Figure4.9 shows misuse cases of access to private or secret information, eavesdropping and taking control of the user's terminal. An explanation appears in Table 4.6.

Table 4.6: Misuse case diagram: Information Disclosure

Threat	**Description**	**Motivation**	**Mitigation**
Access private or secret data	If the application/service stores private or secret data, an attacker can try to access such data by exploiting weaknesses in access control mechanisms, vulnerabilities in the application/system or by attacking the storage media.	The attacker tries to access information that is private, or secret. This could give him private information that the system is manipulating, in this case medical related. Or he can try to access to secret information for enable him to gain knowledge to defect security mechanisms.	Encrypt data during storage. Information can be encrypted during storage to make it harder for attackers to get access to the information. When using encryption it is important to choose recognized encryption algorithms and to protect keys. Enforce proper access control. Control who gets access to resources, and make sure access control mechanisms cannot be circumvented. Use strong authentication. Do not rely on password based authentication, but rather use two-factor authentication by e.g. also using digital certificates or one time passwords.

Continues on next page

Table 4.6 – *Continued from previous page*

Threat	Description	Motivation	Mitigation
Eavesdrop	With the right tools, all network data can be read by an attacker.	The attacker try to monitor the communication between services, or services and users. The motivation is to gain access to the information during transmission instead of need to bypass authentication and authorization mechanism to read the stored data. Other propose is to analyze queries and responses to extrapolate data. Also analyzing communication enables the attacker to gain more knowledge, even capture some credentials, which can use for future attacks.	Encrypt data in transit. By encrypting data in transit it gets harder for attackers to get access to the information by eavesdropping. For encryption to be effective it is however also important to protect keys and to to use encryption algorithms that are well recognized.
Take control over user terminal	Devices such as laptops, USB pens and cell phone often contain confidential data that needs protection.	Patient's and User's terminal systems are juicy in terms of information and some times credentials. The attacker can sometimes get the information directly to the producer or the consumer. Those systems are normally configured with less security mechanisms, with loose security policies, therefor more vulnerable to a successful attack.	Despite of the scope of the proposed solution is restricted to the core architecture, some security activities can be made for mitigate this attack vector. E.g. use disc encryption, encrypt the hard disc content of a terminal; enforce user authentication, enforce all users are authenticated and that the authentication procedure cannot be circumvented.

Protecting private and secret data is critical when manipulating this type of information. The functionalities requested challenge the privacy of the data. Therefore, the data stored in the Repository is double cyphered on a file per contribution, without exposing the data structure which could lead to extrapolation of the stored data. If an attacker wants to read the data, he must gain access to the private key of the Broker, the current internal patient private-key stored in the VHCS and gain access to data stored by a direct Repository attack or by eavesdropping on the communication. To minimize the risks of eavesdropping, services

and their components make use of mutual certificate authentication and cypher the data during the communication. Data sent is cyphered by the communication channel, but it is also cyphered by the requester public key or patient internal public key. This mitigates a man-in-the-middle attack, tricking users into accepting the keys blindly, which could enable an attacker to act as a proxy in the middle of two cyphered channels to read the information in plain text. If this attack succeeds, the information that the attacker gained is also cyphered, and so privacy is not compromised.

The critical service is the VHCS which holds the credentials that can give access to the information cyphered to the patient internal public-key. The other architecture services meet the challenges of being deployed in normal service providers, since the risk of information disclosure even from the services and system administrator is mitigated as the information is manipulated through cyphering without access to the keys used. The VHCS service, especially Cryptographic Component, Patient Credentials and Repository Management, should deal with trustworthy providers, as it manipulates cryptographic function, has the credentials for the repository and the patient internal private-key that can decipher the information stored for the patient. But as the information is stored in other services, the attacker has to corrupt more that the VHCS to gain access to the information.

Denial of service is the result of disrupting normal service provision. A generic misuse case is illustrated in Figure 4.10 and explained in Table 4.7.

Table 4.7: Misuse case diagram: Denial of Service

Threat	**Description**	**Motivation**	**Mitigation**
Make service unavailable	Denial of service is the common term for blocking access to network services and resources.	The attacker may want to disrupt the normal functioning of the services or make it unavailable users, not just for not enable to make use of the services but also to easily substitute it by a malicious service. The malicious service could help the attacker to gain access to privilege information, that can be used for deeper attacks.	Restrict number of concurrent requests. Reduce the amount of traffic to the server by using e.g. ingress filtering. Use fault tolerant mechanisms. Utilize clustering and load balancing. Consider replicating parts of your system so that you can handle failure of individual components.

Mechanisms for achieving fault tolerance and load-balancing may mitigate denial of service attacks. To enable correct exploitation of the load-balancing mechanisms, the proposed solution was based on small, simple sessions, and when possible stateless components, enabling them to be easily replicated and concurrently executed, allowing fault tolerance and scalability.

Elevation of privileges relates to an attacker or authorized user's attempt to gain more

privileges. This aspect is explained in Figure 4.11 and Table 4.8.

Table 4.8: Misuse case diagram: Elevation of privileges

Threat	**Description**	**Motivation**	**Mitigation**
Bypass access control	Poorly implemented access controls can be circumvented, e.g. by using SQL injection to comment out the password validation. URL jumping is another attack path.	Systems and applications, enable to have different privileges for users by implementing access control mechanisms. Attacking those mechanisms could enable to gain access access to the system or to escalate privileges that should not be available for that user.	Sanitize input. Enforce proper access control. Control who gets access to resources, and make sure access control mechanisms cannot be circumvented.
Take control over administrator terminal	Attacker gets illegitimate access to an administrator's terminal.	Taking control over the administrator terminal could give access to credentials for manipulate the access control mechanisms, enabling the attacker to gain access to systems and services. This enable the attack to gain access without the need of attack the services, as he will access with valid credentials, likely with special privileges.	Enforce proper access control. Use strong authentication. Do not rely on password based authentication, but rather use two-factor authentication by e.g. also using digital certificates or one time passwords.

Continues on next page

Table 4.8 – *Continued from previous page*

Threat	Description	Motivation	Mitigation
Get access to administrative operations	Attacker gets access to operations only meant for administrators.	The attacker tries to gain access to operations that are intent to be available only to administrators by exploring misconfiguration, weak passwords, or incorrect privilege assignment for manipulate access control mechanisms. Enabling him to execute operation that he can use to his own propose – owning the systems, services and information.	Use strong authentication. Follow principle of least privilege. Make sure users and applications only have access to information and resources that they need. Consider among others access control policies and the privileges of applications. Force strong password. Require that passwords are of sufficient length and that they contain numbers, symbols and upper and lower case letters. Check passwords against dictionaries.

The process of elevating privileges is through tricking the access control mechanism or gaining more and more privileges by exploiting poorly implemented security mechanisms, vulnerabilities in the systems and services that support the proposed solution, to reach administrative operations. The proposed solution makes use of thorough authentication to minimize the risks associated with password issues. The proposed solution follows the principle of the least privilege in all services and components and makes it necessary to corrupt more than one service to gain access to a patient's information in a repository. It even protects from system administrator access to the information that their services have stored, except components in the VHCS. The VHCS components can detect if the information stored has not been tampered with, using digital signing of the information stored.

A threat model was presented, based on the most general attacks and mitigation actions with analysis of the proposed solution. Given the value of the manipulated information, a more comprehensive analysis of attack vectors to gain access to patient information is made, considering the attempt to access the patient's medical information stored in services or capture it during communications.

Considering that all the services and components of the architecture require mutual authentication and use certificates issued by an internal Certificate Authority the attacker needs to corrupt the CA to create a certificate enabling him to communicate with the other services. Then he can launch other types of attacks on other services or components. This attack is very challenging as the online PKI services are mostly related to the lists of revoked certificates and validation, and even for those services he needs a valid certificate. The certificate is created in an offline node as guidelines suggest. Therefore the attacker normally chooses an easier approach trying to attack more exposed services, such as the VHCS, Broker and user stations, as earlier discussed.

The normal behavior of the solution is to have users requesting actions from the Broker. These actions imply access to the Repository and further manipulation on the VHCS. Capturing traffic between these three services has interest to the attacker as he can try to eavesdrop on communication between these three services. The attack tree is shown in Figure 4.12. He first needs to break the network encryption or make a man-in-the-middle attack. Nonetheless, the attacker will still have access to cyphered information, because the data in transit is cyphered and not only the channel. If he captured it during a request to the Broker, the attacker needs to gain access to private keys. In communication between the Broker and the VHCS when storing, the attacker needs the patient internal private key stored in the VHCS in the Patient Credential component. When retrieving, the attacker needs the user's private key owned by the user to control his physical access. Another approach is to try to capture the traffic between the Broker and the User. When the user is storing, the attacker needs the patient internal private key. When retrieving, the attacker needs the user private key to decipher the data. Another possibility is to try to capture and decipher the data between the Broker and the Repository. Here, the attacker needs access to the Broker's private key and the patient internal private key.

A direct attack on the service Broker manipulating requests is also possible. If successful, the attacker could gain access to the Broker key used for cyphering data to and from the Repository. However, this would not give access to the information, as it is cyphered to the patient internal public-key or the user public-key. Even queries are encrypted and so protected from attacker analysis. The attacker could try to change the registry, creating a new entry for him, or associate his certificate with another provider already in the system. In the first approach, the attacker would try to create a false account, to deceive patients into giving him access to their medical data. In the second one, the attacker would try to spoof the identity of a provider that patients had already given access to their medical data. But as the Broker will check with a valid and trusted CA, the system should detect that the information on the certificate or the CA signature is not correct, and therefore block access to whoever uses that certificate.

The attacker could also try to change the access control defined by a patient by manipulating the information on the Policy Management in the VHCS. The system stores permissions with a signature generated by the authentication token the patient is using, so tampering can be detected. The attacker could only change or create other entries if he gained access to the patient authentication token. But the attacker would probably try to request the Broker for all the patient's information immediately, before the patient requests revocation of the compromised token. Such a query to retrieve all the information can easily trigger some red flags. Therefore, the attacker could create an entry on the policy to enable him to use the user normally by requesting information from the Broker. The attacker would need to have created a fake account with the Broker earlier, which is very difficult to achieve without being detected. Also, the time-stamps of rule creation give information that the rules were created close to the loss or corruption of the token, thus, enabling revocation of those rules or requesting the patient to revalidate those rules with the new token.

A direct attack on the VHCS can also be made, to get access to information from some of the components of that service. It is very difficult to modify the information present in this service as it uses a digital signature on most of the information stored in its components. The attacker could gain access to the patient internal private key and the Repository credentials the Broker uses to access the Repository, but even with this information he will need the key the Broker uses to cypher the data in the repository. The attacker also needs to corrupt the

Broker to be able to gain access to information in plain text.

Another approach is to attack the servers which support the services. This is also challenging, as even if the attacker could bypass firewalls, authentication and authorization mechanisms, gaining administrative privileges on the server, the information stored in them is cyphered for the patient's public-key. Not even the data structure can be analyzed, because in the Repository the data and structure are stored in cyphered data bulks, and on the Indexer too. If the attacker had somehow got the patient internal private key, he could try to corrupt the Indexer, attacking the server that supports the service, because the service accepts only connections from services that have a valid certificate. Therefore, a direct attack on the underlying server could be the attacker's preferred approach.

The service Accounter receives all the logs from the services, ensuring privacy by cyphering the log information created by other services to the corresponding public internal patient key. The service accepts connections from all services with a valid internal certificate, but only to append information. Therefore, the system can verify the correct behaviour of all services and users, and using continuous accounting, could trigger security actions to mitigate possible abuse, as it can detect use of the same user certificate from different locations, incorrect signature on policies, erroneous behaviour of actors, and more.

4.4 Summary

This chapter presented a functionality analysis of the proposed architecture to verify that it answers to the requirements. Also, proposed a methodology to inspect the security of the proposed architecture. This step enabled to verify if every request security proprieties were in place and that no structural design decision that can compromise future development of the security mechanisms was made. Considering this a critical step when proposing a solution, we defend that this approach can be used to verify other architectures. The process can be summarised as follows:

- To identify the functions and the nodes which provide them
- For each STRIDE vector use the misuse cases that make sense to the architecture, using SVRS models, combined databases with past security data and own security expertise
- To construct the threat model using that knowledge and verify how the architect will behave with those misuse cases
- To complement with the table that analyses each misuse case, objective and mitigation activities
- For misuse cases without mitigation strategies or complex ones, attack trees can be created to better understand how the attack is performed and its extension. Enabling also to understand how to mitigate it
- If the attack vector represents very high risk, the architecture may need to be changed to provide mitigation to this attack, if so the analysis should be restarted, since anew security functionality can also lead to a new vector of attack
- When all the desired mitigations are in place and verified, or the risk can be considered acceptable, the analysis can stop

One of the big challenges found is the privacy protection of the information during transmission, storing and processing. As providers are migrating to solutions based on cloud services the extension of a break, malicious users and administrators bring a huge challenge to maintaining information's privacy and at the same time enable the collaboration. Another important attack vector is the analysis of data structure or the queries made to a database to extrapolate the information store on databases or files, even with cyphered content.

The mitigation techniques used for this type of information, despite of the normal controls, were based on the minimal functionality of each component, cyphering storage and indexation. The access to the clinical information required more than one service to interact and grant access. Hence, the information is manipulated in cyphered bulks as much as possible, even in query functions. Besides, the use of a central service for continuous accounting of all services enables the correlation between actions and requests, thus, enabling to take preventive measures with more accuracy to protect the information and the infrastructure.

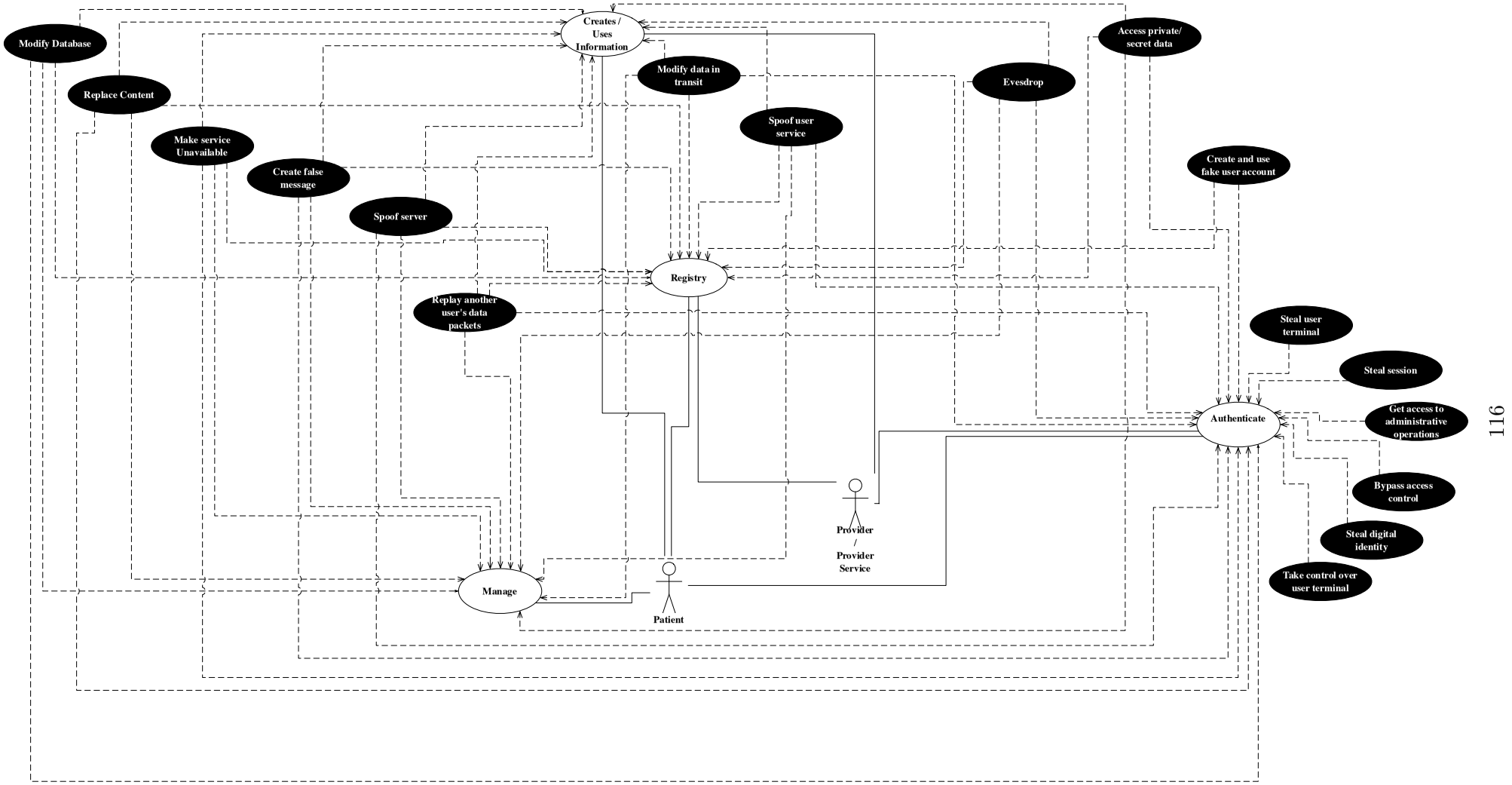

Figure 4.5: Misuse case high level

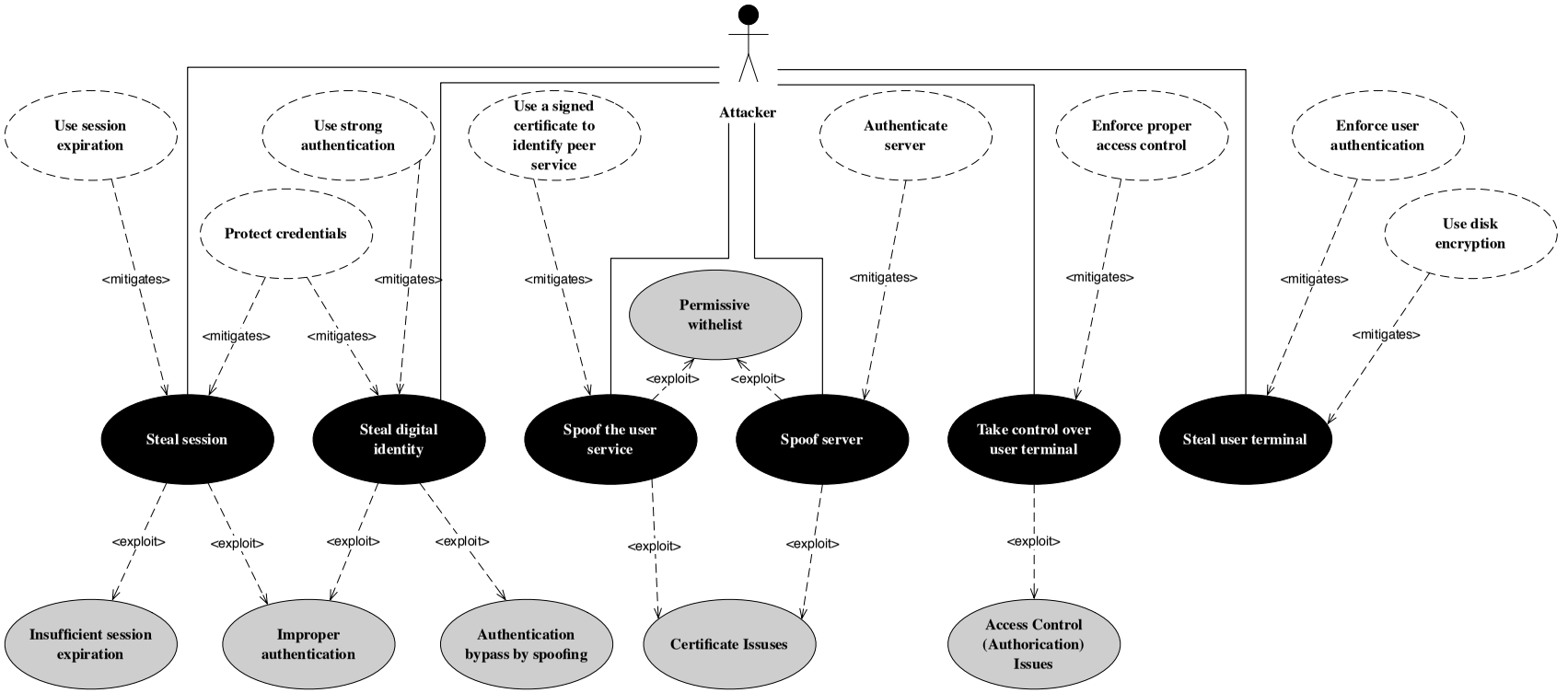

Figure 4.6: Misuse case for spoofing attacks

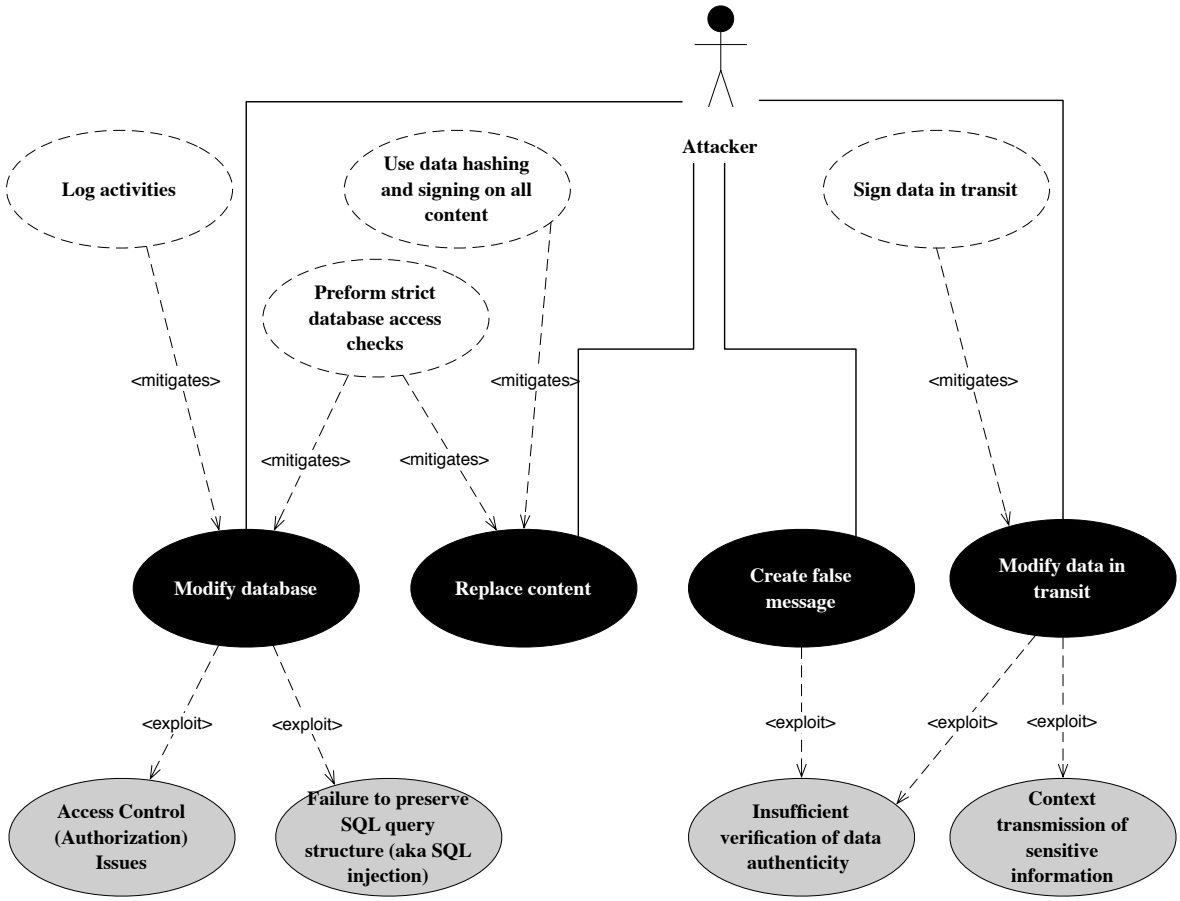

Figure 4.7: Misuse case for tampering attacks

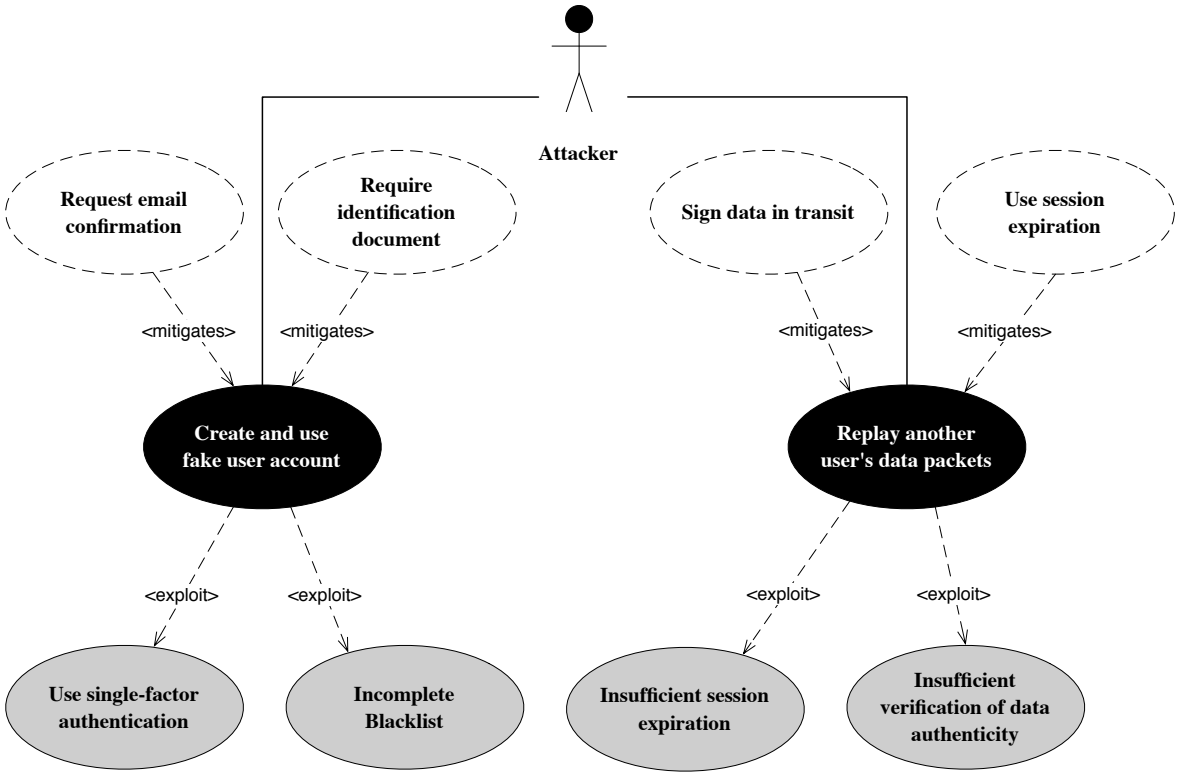

Figure 4.8: Misuse case for repudiation attacks

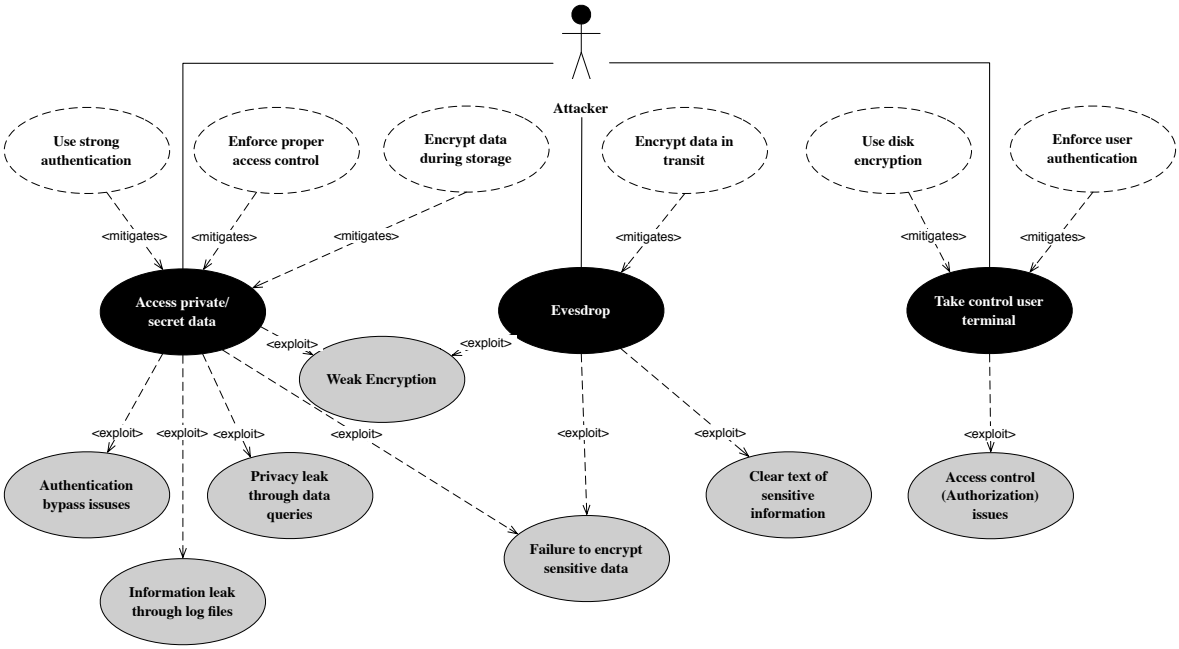

Figure 4.9: Misuse case for information disclosure attacks

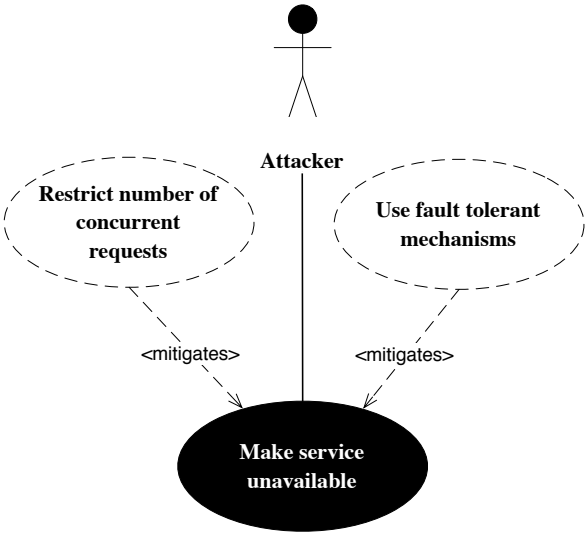

Figure 4.10: Misuse case for denial of service attacks

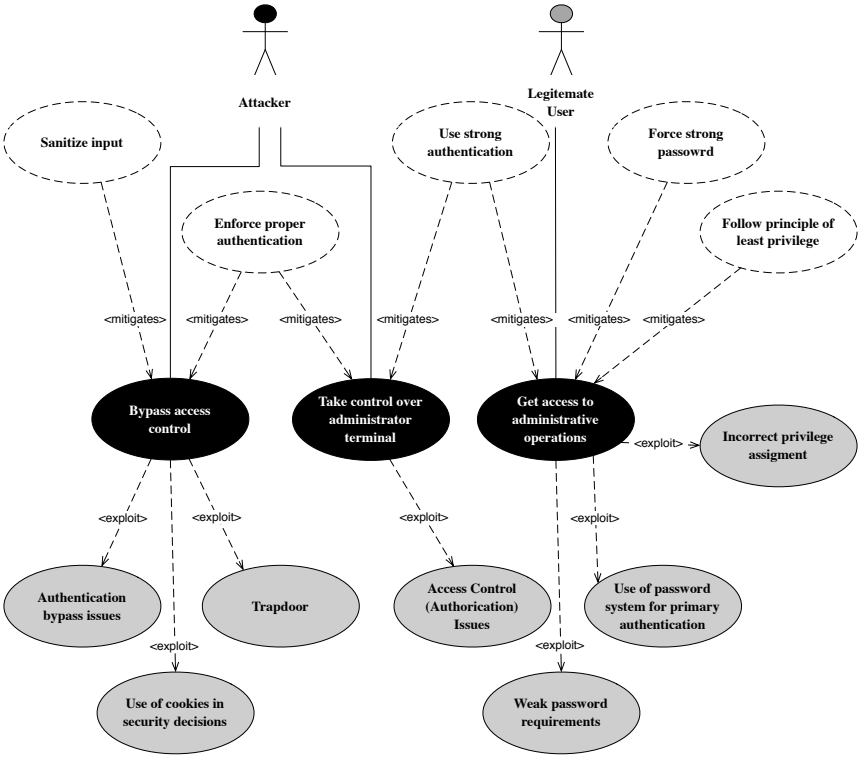

Figure 4.11: Misuse case for elevation of privileges attacks

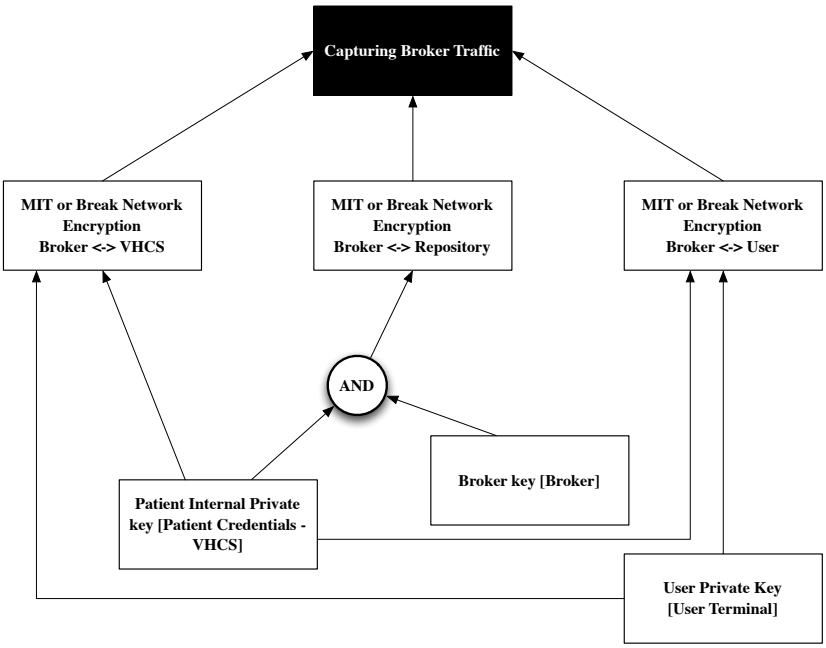

Figure 4.12: Attack tree - Capturing Broker Traffic

Chapter 5

Conclusions and Future Work

This dissertation started with the incursion in the area of medical informatics, specifically in the Electronic Health Record. Electronic Health Records for mobile citizens was the proposed challenge. As such, the initial research focused on patient's mobility and the problem of scattered information. This reality fostered the study of a collaborative solution, managed by the patient, involving all actors, clinical or not, to achieve an updated record, composed by clinical and complementary information. Most of the available solutions does not enable actors to have an active collaboration where every actor can contribute and benefit with the collaborative approach. The lack of access by all actors hinders the massification of the Electronic Health Records use, which limits improvements on health-care services and operational costs. Also, supporting the collaboration of complementary health-care services enable to have richer records which can help professionals keep a broad patient health monitoring, providing integrated care solutions (specially designed for each citizen) based on clinical and non-clinical information that resides on the record. A solution that enables the creation of new services, new providers, easy association and a secure international collaboration between actors managed by citizens will push further the use of Electronic Health Records.

The research of solutions will have to cope with the collaboration goal emanated challenges, such as freedom, openness, evolution, trust and security. Targeting those challenges, we have studied solutions and developed a secure architecture to allow all actors' collaboration, fostering patient empowerment and conciliating clinical integrity with third party access. A brief description follows of how challenges were addressed.

Freedom is related with the patients' liberty of choice and his empowerment. He should be able to choose his provider, to manage access control to his record and to contribute to it, without depending on service agreements between providers or by the technology used. The creation of an open architecture, using industry standards enables any provider to develop their tools and services upon the architecture. Moreover, it enables patients to choose their repository provider. The coexistence of information produced by clinical staff and third party is address by the concept of hybrid Electronic Health Records, which combine the clinical trust of EHR and the openness with patient control of PHR.

The openness is achieved through open standard framework, such as OpenEHR, specifically by using archetypes for data format which is serialized to XML, moreover, a service oriented architecture with public available interfaces enables providers and enterprises to develop their products.

The trust between actors and integrity of data is achieved by the use of certificates on

electronic identification cards or signed by trusted Certificate Authorities. It also enables users auto-enrolment with strong authentication after prior validation of authorities. OpenEHR archetypes helps to make some semantic validation of the stored data, combined with digital signature of the stored information the integrity is protected.

Systems that manipulate and store medical information have to deal with lifelong storage challenges and need to support evolution. OpenEHR archetypes enable the coexistence of different versions of the same archetype, enabling services to use the correct one for the data that is on the repository. Thus, new archetypes can be developed for supporting new services, to incorporate new types of data on the repository and to support new procedures.

The security issues were incorporated since the design phase. Strong authentication is made using certificates, the services make mutual authentication, and only trusted Certificate Authorities are valid to be used by patients and providers. Internal services use an internal CA. The information is always manipulated and stored cyphered. The only service that temporary manipulates the information in plain text is the VHCS and only at the cryptographic and reassembler component. Data and structure are stored together in an XML file, protecting from analysing the structure for extrapolating data contents. Privacy is ensured also from the services administrators, with exception of the cryptographic and reassembler component of the VHCS. The tampering of patients' policies and actors' descriptions can also be detected because they are signed using the actors' authentication tokens, whose private keys reside outside the architecture and only the actors have access to. The data integrity of the logs and patient clinical information can also be verified due to the use of digital signatures.

Resuming the security validation, the attack vector for gaining access to a patient's clinical information is very expensive. An attacker needs to corrupt patient's internal private key, the broker internal private, repository credentials and deceive the Accounter service simultaneously. The most critical data stored in the architecture are the Patient Credentials on the VHCS, since it contains the internal private key for each patient. The Accounter can detect abnormal services requests since it receives information from all the services, enabling to have a global view of the requests, with traceability to the requester actor.

The proposed architecture does not have the intention to substitute systems and records already being used in hospitals and clinics. Those systems can continue to be used on their organizations, but should enable the generation of reports to the architecture and when allowed by the patient can access to the patient controlled collaborative record.

5.1 Main Results

This dissertation contributed mainly with a novel and secure approach for dealing with the problem of creating and maintaining an updated record that can cope with mobility. During the initial research diverse approaches where studied, but in the end all of them where oriented to a specific set of requirements. Of course, when the requirements change or evolve the solutions normally tend to fail. When figuring out a solution that could cope with the mobility issues and that empowers the patient, we did not try to change the normal way of working. If providers use their custom configured systems for manipulate records, and have their federations for enabling sharing they can maintain their workflow. Our approach only requires the creation of resumes to be submitted to our service architecture, where the patient can control the access by others and enable open collaboration of all the actors.

The solution can aggregate information from the medical staff, patient and third parties. The aggregation is supported by hybrid Electronic Health Records that are based on the combination of Electronic Health Records and Personal Health Records approaches. It incorporates the clinical data integrity of Electronic Health Records with the patient empowerment of the Personal Health Records. The patient can control the access of all the actors he wants and the medical staff can trust in the clinical information on that repository, enabling to have a richer repository with complementary information that could enable better and differentiated care. Every actor that the patient wants can have access to his record, in a simple way, without the bureaucratic burden of communication agreements between the different providers, enabling a wider concept of collaboration, where the limitation is made by access to Internet, open and standard tools and patient consent.

The success of the previous approach depends on a secure architecture that provides the functionalities needed to support this type of records. The more open and secure the architecture, the more actors are expected to contribute, enriching the patient repository. During this phase, the necessary functionality was combined with security requirements and use of open standards in industry technology. This openness can enable more players to develop tools and services upon this base architecture. The security concerns modeled the way of managing the data and the collaboration. Functionality was divided in different services and components, where each component was only local view of data and needs collaboration of other components and services to be able to respond to actor's requests. This approach allows the creation of a more secure solution based on the principle of minimal functionality and privileges.

For a secure and robust solution, security should be involved in all phases. This dissertation also contributed to the creation of a threat model for a service architecture that deals with patient clinical data. The treat model followed the STRIDE approach for creating a misuse case for the architecture. It enabled reasoning on how the architecture would respond under the considered attacks.

This work contributed to the body of knowledge by proposing an innovative approach of collaboration of all actors to create and maintain patient's health record – the hybrid Electronic Health Records. It presented a secure architecture for manipulate the records. Furthermore, a security analysis of the full proposal using the threat model was made.

5.2 Future Work

The natural step that follows is further developing and implementing the proposed architecture.

When all the components have been fully implemented, we expect to make performance testing, verifying individual components' times. Despite the fact that the architecture had been modeled to enable the use of high availability mechanisms, some measurements can help to determine where to start to optimize the solution. The analysis can indicate where further research can be applied to optimize the response time of the architecture, especially when retrieving information. One of the critical performance issues can be related with the heavy use of cryptography. There is also a big performance difference between symmetric and asymmetric cryptography. A hybrid solution can bring advantages when dealing with large data block [114, 115]. The proposed solution makes heavy use of public key cryptography, the information produced is related with a patient, therefore cyphered for his public

key. This way the different actors and services do not share a symmetric key, enforcing that only patient internal private key can be used to open the created information. On storing, the data to be cyphered tend to be small. On the retrieving process, if the response is composed by various individual contributions, the contributions are deciphered using the patient internal key, reassembled and cyphered for the requester actor's token public key. This can be a procedure that may gain of the hybrid approach, the public key cryptography can negotiate to negotiate a symmetric key for the VHCS to be used for cyphering the response to the requester. This can bring other attack vectors and should be measured to verify if it changes the overall response time. Concerning that the architecture uses web services, for making a request between services, the use of the public key cryptography enables to make the request in just one instruction. The need to make more requests between requester–Broker–VHCS for negotiating a symmetric key, considering the small volume of data to be cyphered, could be overkill. Hence, dedicated hardware is being developed for accelerating cryptographic functions [116], using graphic processors [117], and pipelining/parallelism in crypto engines [115], which decrease the time spent in cryptographic functions. Even then, symmetric crypto should be faster than asymmetric, in the cyphering/deciphering processes, but the values may start to be negligible when considering the impact in the overall performance. Another point that should be considered is the study of the applicability of other asymmetric key techniques, such as Elliptic Curve Cryptography. As they are considered to enable a faster method of encryption when compared with current standard public key cryptography algorithms of RSA [118, 119, 120].

A full assessment should be made using test data and actors. To create the test data the approach, if no better test data appear, is to generate random data based on the constrains of the archetypes as it was used for OpenEHR repository based on web services and native XML databases.

Another line of research is to study the deployment of the proposed architecture on the cloud. We believe that it can be done in a smooth and efficient way, as the architecture is service oriented, it can be easily deployed in such a scenario. The security functionalities of the solution enable more trust regarding privacy, confidentiality and integrity of the information. The full architecture can be deployed in a hybrid cloud environment where the most critical component – the VHCS, should be implemented in a private cloud and the others could be deployed in public ones. The change of the deployment scenario may slightly change requirements pushing further changes on the architecture. During this study, a new threat model analysis should be made to verify how it would cope with some attack vectors.

Solutions specifically orientated to the sharing of medical imaging are being researched, within our group. A line of investigation is the complementariness of combining solutions. Starting with the proposal of an external service that when creating the contribution, can push the imaging file to the specific store solution, it generates a URL for storing with the contribution to enable future retrieving if necessary.

One of the contributions of this thesis is the proposal of a solution that can work as a back-end to enable to collaboration of all actors the patient would like to involve in the production and use of his clinical and clinical related data. Therefore, this architecture can support the creation of new services. Challenges need to be addressed, such as systems already in place need to create their submission systems, with the capability of *translation* to the OpenEHR format. If needed, new archetypes should be proposed and agreed, and maintaining the semantic of the contribution is a special challenge. Another challenging issue is the automatic extraction of the metadata, of the record that is being stored, if we want

a less naive solution than using the most representative fields of the archetype. Another interesting problem is the automatically classification of sensitivity level of the information that is being uploaded.

Considering that this solution will fail when connectivity fails, a solution that will enable the patient to have a synchronized replica, or a snapshot, of the information in the repository in an external mass storage can be helpful in same cases. The approach based on a self-managed and contained Personal Health Record is also being studied in our research group. It makes use of OpenEHR as data format, with different containers on a usb mass storage device. Therefore information can be created on the storage device, later be uploaded to the architecture, and the pen could receive information from the architecture. One could backup the other. Even so, a service to manage the synching mechanism needs to be analyzed.

The openness and the use of standard information technologies tools enable any actor to create services to make use of the proposed architecture. Providers of wearable monitoring gear, remote monitoring, sports activity monitoring and other sensor bases can provide capabilities to the base station to upload the information directly to the patient repository. Also, producer/consumer services can be created such as, patient annotation service, patient logbook, service for monitoring the diet of a patient and his evolution. Others can be patient alert systems, e.g. for medication, treatments, consultation and other alerts.

The capability of aggregating information from so many sources in a structured, trusted and semantically enforced record will have huge value. Hence, another line of research is the creation of a service that would enable a researcher to make queries and retrieve information from the patient repository, if duly authorized by the patient. A researcher would gain access to a very rich repository that can enable them to find new relations, analyze procedures, treatments, detect anomalous reactions and detect public health problems in a proactive scenario. Anonymization of the patient information before sending it to researchers is one big challenge. There are some ways for de-identification, some are already in place as the identification among the identification number on the records and the demographic information is only made by the VHCS. Although, for some type of stored information the de-identification is not sufficient to protect the patient from being identified. Thus, the anonymization tries to go further to stop the ability to identify the patient. This approach depends of the type of data stored, as in the repository different types of information coexist and the challenge increases.

Existing and future services will enrich the ecosystem, with new type of information or new and innovative ways of manipulating and using the information already stored, enabling the patient to maintain a full updated collaborative electronic health record that can evolve, keeping up with new requirements.

Bibliography

[1] R. Zhang and L. Liu. Security Models and Requirements for Healthcare Application Clouds. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 268–275. IEEE, 2010.

[2] A. Sunyaev, JM Leimeister, A. Schweiger, and H. Krcmar. It-standards and standardization approaches in healthcare. *Encyclopedia of Healthcare Information Systems. Editors: Wickramasinghe, N.; Geisler, Publisher: Idea Group*, 2008.

[3] HL7. Hl7 standards timeline. `http://www.hl7.org/documentcenter/public_temp_D67F8EC2-1C23-BA17-0C34F13A4B0DD1EE/training/IntroToHL7/data/downloads/hl7_standards_timeline.pdf`, 07 2011.

[4] Digital Imaging and Communications in Medicine (DICOM). Digital imaging and communications in medicine (dicom) part 18: Web access to dicom persistent objects (wado). Technical report, NEMA Publications, 2009.

[5] M. Eichelberg, T. Aden, J. Riesmeier, A. Dogac, and GB Laleci. Electronic health record standards–a brief overview. In *4th international conference on information and communications technology, Cairo, Egypt*. Citeseer, 2006.

[6] T. Beale, S. Heard, D. Kalra, and D. Lloyd. Openehr architecture: Architecture overview. *OpenEHR Foundation. Accessed October*, 2007.

[7] D.A. Clunie. *DICOM structured reporting*. PixelMed Publishing, 2000.

[8] Linda Maria Velte. Repositório de registos electrónicos de saúde baseado em openehr. Master's thesis, Universidade de Aveiro - Departamento de Electrónica e Telecomunicações, 2011.

[9] epSOS. epsos technical aspects. Technical report, epSOS, 2012.

[10] EN 13606-4:2007(E). *Part 4: Health informatics - Electronic health record communication – Security*. CEN, 2007.

[11] S. Hernan, S. Lambert, T. Ostwald, and A. Shostack. Threat modeling-uncover security design flaws using the stride approach. *MSDN Magazine-Louisville*, pages 68–75, 2006.

[12] ESI, SINTEF, LIU, and TXT. Detecting known security vulnerabilities from within design and development tools - d1. 4 final shields approach guide. Technical report, SHIELDS Project `http://www.shields-project.eu/`, 2010.

[13] G. Eysenbach. Medicine 2.0: social networking, collaboration, participation, apomediation, and openness. *Journal of Medical Internet Research*, 10(3), 2008.

[14] J.G. Hodge Jr, L.O. Gostin, and P.D. Jacobson. Legal issues concerning electronic health information: privacy, quality, and liability. *Jama*, 282(15):1466, 1999.

[15] A. Shabo. A global socio-economic-medico-legal model for the sustainability of longitudinal electronic health records-part 2. *Methods of information in medicine*, 45(3):240, 2006.

[16] M.K. Abd Ghani, R.K. Bali, R.N.G. Naguib, and I.M. Marshall. Electronic health records approaches and challenges: a comparison between Malaysia and four East Asian countries. *International Journal of Electronic Healthcare*, 4(1):78–104, 2008.

[17] I. Román, LM Roa, J. Reina-Tosina, and G. Madinabeitia. Demographic management in a federated healthcare environment. *International Journal of Medical Informatics*, 2006.

[18] E. Coiera. *Guide to health informatics*. Arnold London, 2003.

[19] WJ Pories. Is the medical record dangerous to our health? *NC Med J*, 51(1):47–55, 1990.

[20] Aykut M. Uslu and Jürgen Stausberg. Value of the electronic patient record: An analysis of the literature. *Journal of Biomedical Informatics*, 41(4):675 – 682, 2008.

[21] HIMSS. Himss ehr definition. `http://www.himss.org/ASP/topics_ehr.asp`, July 2010 (accessed July 19, 2010).

[22] D. Garets and M. Davis. Electronic medical records vs. electronic health records: yes, there is a difference. *A HIMSS Analytics White Paper. Chicago, IL: HIMSS Analytics*, 2005.

[23] Technical Committee ISO/TC 215. Health informatics — electronic health record — definition, scope, and context - iso/tr 20514:2005(e). Technical report, International Organization for Standardization, 2005.

[24] NCRR. Electronic Health Records Overview. Technical report, MITRE, April 2006.

[25] The Lancet. Electronic health records. *The Lancet*, 371(9630):2058–2058, June 2008.

[26] HIMSS. Himss phr definition. `http://www.himss.org/ASP/topics_phr.asp`, 2010 (accessed July 19, 2010).

[27] J.C. Santos, Tiago Pedrosa, Carlos Costa, and J.L. Oliveira. Modelling a portable personal health record. In *HealthInf 2010 Proceedings*, 2010.

[28] M. Eichelberg, T. Aden, J. Riesmeier, A. Dogac, and G.B. Laleci. A survey and analysis of electronic healthcare record standards. *ACM Computing Surveys (CSUR)*, 37(4):277–315, 2005.

[29] D. Kalra. Electronic health record standards. *IMIA yearbook of medical informatics*, 2006:136–144, 2006.

[30] R.H. Dolin, L. Alschuler, S. Boyer, C. Beebe, F.M. Behlen, P.V. Biron, and A. Shabo Shvo. Hl7 clinical document architecture, release 2. *Journal of the American Medical Informatics Association*, 13(1):30, 2006.

[31] George W Beeler. Hl7 version 3–an object-oriented methodology for collaborative standards development. *International Journal of Medical Informatics*, 48(1-3):151 – 161, 1998.

[32] G.V. Koutelakis and D.K. Lymperopoulos. Pacs through web compatible with dicom standard and wado service: advantages and implementation. In *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*, pages 2601–2605. IEEE, 2006.

[33] Sanromà, M.B. and Mateu, A.V. and i Oliveras, K.G. and Universitat Rovira i Virgili. Departament d'Enginyeria Informàtica. *Survey of Electronic Health Record Standards*. Universitat Rovira i Virgili. Departament d'Enginyeria Informàtica, 2006.

[34] T. Beale, S. Heard, D. Kalra, and D. Lloyd. Archetype definition language (adl). *OpenEHR Foundation*, 2005.

[35] T. Beale, S. Heard, D. Kalra, and D. Lloyd. Archetype query language (aql). *OpenEHR Foundation*, 2005.

[36] R. Noumeir. Integrating the healthcare enterprise process. *International Journal of Healthcare Technology and Management*, 9(2):167–180, 2008.

[37] T. Aden and M. Eichelberg. Cross-enterprise search and access to clinical information based on the ihe retrieve information for display. *International Congress Series*, 1281:986–991, 2005.

[38] A. Dogac, V. Bicer, A. Okcan. Collaborative business process support in ihe xds through business process. In *Proceedins of the 22nd International Conference on Data Engineering*. IEEE Computer Society, 2006.

[39] A. Dogac, T. Namli, A. Okcan, G. Laleci, Y. Kabak, and M. Eichelberg. Key issues of technical interoperability solutions in ehealth. In *Proceedings of eHealth 2006 High Level Conference Exhibition*, 2006.

[40] The European Economic and Social Committee and the comittee of regions. Final report on the implementation of the commission's action plan for skills and mobility com(2002) 72 final. Technical report, Commission of the European Communities, 2007.

[41] D. Ferreira Polónia, C. Costa, and JL Oliveira. Architecture evaluation for the implementation of a regional integrated electronic health record. Connecting Medical Informatics and Bio-Informatics—Proceedings of MIE2005—The XIXth International Congress of the European Federation for Medical Informatics. Geneva: IOS Press, 2005.

[42] Carlos Costa, José Luís Oliveira, Augusto Silva, Vasco Gama Ribeiro. A new concept for an integrated Healthcare Access Model. *Studies in health technology and informatics*, 95:101, 2003.

[43] JS Wimalasiri, P. Ray, and CS Wilson. Security of electronic health records based on Web services. *Enterprise networking and Computing in Healthcare Industry, 2005. HEALTHCOM 2005. Proceedings of 7th International Workshop on*, pages 91–95, 2005.

[44] M. Hori and M. Ohashi. Applying XML Web Services into Health Care Management. *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*, pages 155a–155a, 2005.

[45] Tiago Pedrosa, Carlos Costa, R.P. Lopes, and J.L. Oliveira. Virtual health card system. *Inforum 2009*, 2009.

[46] Tiago Pedrosa, R.P. Lopes, J.C. Santos, Carlos Costa, and J.L. Oliveira. Towards an EHR architecture for mobile citizens. In *HealthInf 2010 Proceedings*, pages 288–293, Valencia, Spain, 2010.

[47] Pradeep Ray and Jaminda Wimalasiri. The need for technical solutions for maintaining the privacy of EHR. In *Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, volume 1, page 4686, 2006.

[48] L. Seitz, J.M. Pierson, and L. Brunie. Encrypted storage of medical data on a grid. *Methods of Information in Medicine*, 44(2):198—-201, 2005.

[49] H. Wang and L.V.S. Lakshmanan. Efficient secure query evaluation over encrypted XML databases. In *Proceedings of the 32nd international conference on Very large data bases*, page 138. VLDB Endowment, 2006.

[50] Aministração Central do Sistema de Saúde. Rse – registo de saúde electrónico - r1: Documento de estado da arte. Technical report, Ministério da Saúde, 2009.

[51] Administração Central do Sistema de Saúde. Rse – registo de saúde electrónico - r2a: Orientações para especificação funcional e técnica do sistema de rse. Technical report, Ministério da Saúde, 2009.

[52] epSOS. Safe access to medical data abroad: The epsos services. Technical report, epSOS, 2012.

[53] epSOS. D3.3.2 final epsos system technical specification. Technical report, epSOS, 2010.

[54] epSOS. Work package 3.5 - semantic services appendix g - cen en13606 technical specifications. Technical report, epSOS, 2009.

[55] Lei nº 67/98, 26 Outubro 1998.

[56] Lei n.º 48/90, de 2l de agosto lei de bases da saúde, Agosto 1990.

[57] Código deontológico da ordem dos médicos.

[58] Lei 12/2005 - informação genética e informação de saúde, January 2005.

[59] Council of Europe Committee of Ministers. Recomentation on the protection of medical data. Available from: `https://wcd.coe.int/wcd/ViewDoc.jsp?Ref=ExpRec(97)5&Language=lanEnglish&Ver=original&Site=CM&BackColorInternet=DBDCF2&BackColorIntranet=FDC864&BackColorLogged=FDC864`, 1997.

[60] Council of Europe Committee of Ministers. On the impact of information technologies on health care – the patient and internet, 2004.

[61] D.M. D'Alessandro and N.P. Dosa. Empowering children and families with information technology. *Archives of Pediatrics and Adolescent Medicine*, 155(10):1131, 2001.

[62] Miller SoM. Health insurance portability and accountability act of 1996 (hipaa). Available `from:privacy.med.miami.edu/glossary/xd_hipaa.htm`, 2005.

[63] C.R. Dalton. The NHS as a proving ground for cryptosystems. *Information Security Technical Report*, 8(3):73–88, 2003.

[64] J. Wainer, CJR Campos, MDU Salinas, and D. Sigulem. Security requirements for a lifelong electronic health record system: An opinion. *The Open Medical Informatics Journal*, 2:160, 2008.

[65] FH France and PN Gaunt. The need for security–a clinical view. *International journal of bio-medical computing*, 35:189, 1994.

[66] R.C. Barrows and P.D. Clayton. Privacy, confidentiality, and electronic medical records. *Journal of the American Medical Informatics Association*, 3(2):139, 1996.

[67] A. Ferreira, R. Cruz-Correia, L. Antunes, P. Farinha, E. Oliveira-Palhares, D.W. Chadwick, and A. Costa-Pereira. How to break access control in a controlled manner. In *Computer-Based Medical Systems, 2006. CBMS 2006. 19th IEEE International Symposium on*, pages 847–854. IEEE, 2006.

[68] Javier López-Muñoz Paloma García-López, Stefanos Gritzalis. Monograph: Standardization for ict security, 2005.

[69] ISO 22600-1:2006(E). *Part 1: Health Informatics - Privileges management and policy management – Overview and policy management.* ISO, Geneva, Switzerland.

[70] ISO 22600-2:2006(E). *Part 2: Health Informatics - Privileges management and policy management – Formal Models.* ISO, Geneva, Switzerland.

[71] William Stallings. *Network Security Essentials, Applications and Standards.* Prentice Hall, 2003.

[72] L. O'Gorman. Comparing passwords, tokens, and biometrics for user authentication. *Proceedings of the IEEE*, 91(12):2021–2040, 2003.

[73] A. Jain, L. Hong, and S. Pankanti. Biometric identification. *Communications of the ACM*, 43(2):90–98, 2000.

[74] N.K. Ratha, J.H. Connell, and R.M. Bolle. Enhancing security and privacy in biometrics-based authentication systems. *IBM Systems Journal*, 40(3):614–634, 2001.

[75] Carlos Costa. *Concepção, desenvolvimento e avaliação de um modelo integrado de acesso a registos clínicos electrónicos.* PhD thesis, Universidade de Aveiro - Departamento de Electrónica e Telecomunicações, 2004.

[76] J.J. Shen, C.W. Lin, and M.S. Hwang. A modified remote user authentication scheme using smart cards. *Consumer Electronics, IEEE Transactions on*, 49(2):414–416, 2003.

[77] H.Y. Chien, J.K. Jan, and Y.M. Tseng. An Efficient and Practical Solution to Remote Authentication: Smart Card. *Computers & Security*, 21(4):372–375, 2002.

[78] R. Santos, M.E. Correia, and L. Antunes. Securing a health information system with a government issued digital identification card. In *Security Technology, 2008. ICCST 2008. 42nd Annual IEEE International Carnahan Conference on*, pages 135–141. IEEE.

[79] R. Santos, M.E. Correia, and L. Antunes. Securing a health information system with a government issued digital identification card. In *Security Technology, 2008. ICCST 2008. 42nd Annual IEEE International Carnahan Conference on*, pages 135 –141, oct. 2008.

[80] Qamar Munawer Sylvia Osborn, Ravi Sandhu. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Transactions on Information and System Security (TISSEC)*, 3(2):85–106, May 2000.

[81] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-Based Access Control Models. 1996.

[82] S. Osborn. Mandatory access control and role-based access control revisited. *Proceedings of the second ACM workshop on Role-based access control*, pages 31–40, 1997.

[83] RS Sandhu. Lattice-based access control models. *Computer*, 26(11):9–19, 1993.

[84] Marek Jawurek. Rsbac - a framework for enhanced linux system security. Conference Seminar "Dependable Distributed Systems" - RWTH Aachen University, 2006.

[85] A. Ott and S. Fischer-Hübner. The 'Rule Set Based Access Control'(RSBAC) Framework for Linux. *Proceedings of the 8th International Linux Kongress.*

[86] A. Ott. Rule set based access control (rsbac). In *the Snow Unix Event-unix. nl congress "Reliable Internet", Waardenburg*, 2001.

[87] Alberto Riva, Kenneth D. Mandl, Do Hoon Oh, Daniel J. Nigrin, Atul Butte, Peter Szolovits, Isaac S. Kohane. The personal internetworked notary and guardian. *International Journal of Medical Informatics 62*, pages 27–40, 2001.

[88] M. Evered and S. Bögeholz. A case study in access control requirements for a Health Information System. *Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation-Volume 32*, pages 53–61, 2004.

[89] B.P. Lim, O. Zakaria, and M.K.M. Nor. Role-based Access Control in Kidney Dialysis Information System. *Malaysian Journal of Computer Science*, 14(2):20–25, 2001.

[90] J. Reid, I. Cheong, M. Henricksen, and J. Smith. A Novel Use of RBAC to Protect Privacy in Distributed Health Care Information Systems. *Proceedings of the Eighth Australasian Conference on Information Security and Privacy (ACISP 2003), LNCS*, 2727:403–415, 2003.

[91] José Magalhães Cruz Sara Araújo, João Pascoal Faria. Propostas de melhoria da segurança dos sistemas de informação clínica em portugal. *6th International Conference on the Quality of Information and Communications Technology - QUATIC 2007 -*, 2007.

[92] B. Aboba, D. Harrington, and J. Arkko. Introduction to accounting management. IETF, RFC 2975, 2000, October.

[93] Y. Niu. Soa governance and accounting control. In *Future Computer and Communication (ICFCC), 2010 2nd International Conference on*, volume 3, pages V3–699. IEEE.

[94] D.W.L. Searcy and J.B. Woodroof. Continuous auditing: leveraging technology. *The CPA Journal*, 75(5):46–48, 2003.

[95] H. Ye, S. Chen, F. Gao, Y. He, Q. Li, SY Chen, and A. Xu. Soa-based conceptual model for continuous auditing: A discussion. In *WSEAS International Conference. Proceedings. Mathematics and Computers in Science and Engineering*. World Scientific and Engineering Academy and Society, 2008.

[96] W. Stallings. *Network security essentials: applications and standards*. William Stallings books on computer and data communications technology. Prentice Hall, 2007.

[97] A.J. Menezes, P.C. Van Oorschot, and S.A. Vanstone. *Handbook of applied cryptography*. CRC, 1997.

[98] J. Daemen and V. Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer, 2002.

[99] B. Preneel. A survey of recent developments in cryptographic algorithms for smart cards. *Computer Networks*, 51(9):2223–2233, 2007.

[100] M.J.B. Robshaw. Stream ciphers. *RSA Laboratories, a division of RSA Data Security, Inc*, 1995.

[101] R. Weis and S. Lucks. Cryptographic hash functions. *Recent Results on Cryptanalysis and their Implications on System Security. In: SANE*, 2006.

[102] P. Gauravaram, A. McCullagh, and E. Dawson. Collision attacks on md5 and sha-1: Is this the sword of damocles for electronic commerce. *Information Security Institue (ISI), Queensland University of Technology (QUT), Australia*, 2006.

[103] FIPS PUB 180-3. Secure hash standard (shs) - final. Technical report, Technical report, National Institute of Standards and Technology (NIST), `http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf`, 2008.

[104] W. Diffie and M. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976.

[105] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[106] J. Weise. Public key infrastructure overview. *Sun BluePrints OnLine, August*, 2001.

[107] Sam Heard, Thomas Beale, Gerard Freriks, Angelo Rossi Mori, and Ognian Pishev. Templates and archetypes: how do we know what we are talking about? Available from `http://www.openehr.org/wiki/download/attachments/196620/Templates+and+Archetypes.pdf`, February 2003.

[108] R.L. Jones and A. Rastogi. Secure coding: building security into the software development life cycle. *Information Systems Security*, 13(5):29–39, 2004.

[109] Erkuden Rios, Per Håkon Meland, Shanai Ardi, Alessandra Bagnato, Jostein Jensen, Wissam Mallouli, Fabio Raiteri, Txus Sanchez, Inger Anne Tøndel, and Bachar Wehbi. A qualitative evaluation of model-based security activities for software development. In *European Workshop on Security in Model Driven Architecture 2009 (SEC- MDA 2009)*, 2009.

[110] S. Barnum and A. Sethi. Attack patterns as a knowledge resource for building secure software. In *OMG Software Assurance Workshop: Cigital*, 2007.

[111] B. Lawlor and L. Vu. A survey of techniques for security architecture analysis. Technical report, DTIC Document, 2003.

[112] ESI, SINTEF, LIU, and TXT. Shields - detecting known security vulnerabilities from within design and development tools. Available from: `http://www.shields-project.eu/`, August 2012 (accessed August, 27, 2012).

[113] P.H. Meland, D.G. Spampinato, E. Hagen, ET Baadshaug, KM Krister, and KS Velle. Seamonster: Providing tool support for security modeling. *Norsk informasjonssikkerhetskonferanse, NISK*, 2008.

[114] A. Al Hasib and A.A.M.M. Haque. A comparative study of the performance and security issues of aes and rsa cryptography. In *Convergence and Hybrid Information Technology, 2008. ICCIT'08. Third International Conference on*, volume 2, pages 505–510. IEEE, 2008.

[115] Li Zhao, R. Iyer, S. Makineni, and L. Bhuyan. Anatomy and performance of ssl processing. In *Performance Analysis of Systems and Software, 2005. ISPASS 2005. IEEE International Symposium on*, pages 197 –206, march 2005.

[116] J.K.T. Chang, S. Liu, J.L. Gaudiot, and C. Liu. The performance analysis and hardware acceleration of crypto-computations for enhanced security. In *Dependable Computing (PRDC), 2010 IEEE 16th Pacific Rim International Symposium on*, pages 249–250. IEEE, 2010.

[117] K. Jang, S. Han, S. Han, S. Moon, and K.S. Park. Sslshader: cheap ssl acceleration with commodity processors. In *Proceedings of the 8th USENIX conference on Networked systems design and implementation*, pages 1–1. USENIX Association, 2011.

[118] V. Gupta, S. Gupta, S. Chang, and D. Stebila. Performance analysis of elliptic curve cryptography for ssl. In *Proceedings of the 1st ACM workshop on Wireless security*, pages 87–94. ACM, 2002.

[119] O.M. Creado, X. Wu, Y. Wang, and P.D. Le. Probabilistic encryption–a comparative analysis against rsa and ecc. In *Computer Sciences and Convergence Information Technology, 2009. ICCIT'09. Fourth International Conference on*, pages 1123–1129. IEEE, 2009.

[120] BK Alese, ED Philemon, and SO Falaki. Comparative analysis of public-key encryption schemes. *International Journal of Engineering and Technology*, 2(9), 2012.