Privacy-Preserving Electronic Healthcare with Self-Monitoring Devices using Trusted Execution Environments

by

Seu Man To

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday November 25, 2020 at 3:00 PM.

Student number:	4064976	
Thesis committee:	Prof. dr. ir. R. L. Lagendijk,	TU Delft, chair
	Dr. Z. Erkin,	TU Delft, supervisor
	Dr. S. Roos,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

The adoption of smart wearable devices has been on the rise over the past few years. These wearables are able to track the user's vital signs, making them valuable for use in the healthcare industry. Sharing this information with the user's healthcare provider has the potential to improve medical care by reducing medical misinformation.

Currently, patients in the Netherlands are able to inspect their medical file at each healthcare provider they are attending, but they are unable to inspect all their medical files at one place. Moreover, they are not able to contribute to their own medical files.

In this thesis, we propose the use of Trusted Execution Environments (TEEs) as an extension to the Polymorphic Encryption and Pseudonymisation (PEP) framework. PEP facilitates the exchange of medical data between multiple parties in a privacy-preserving manner. However, PEP suffers from collusion and scalability issues. The Distributed Polymorphic Encryption and Pseudonymisation (Dist-PEP) protocol is an improvement to PEP, and mitigates these issues. To make further improvements, the Distributed Polymorphic Access Management (Dist-PACMAN) protocol has been introduced to handle the access management more securely. By introducing TEEs, we make further improvements to the efficiency en scalability of the protocol. The result is a privacy-preserving framework that can be used to share the information gathered by the wearables with the user's healthcare provider securely and more efficiently. Additionally, the user can keep and inspect their own medical file at a cloud provider.

Preface

My thesis time has been long and stressful, and I am glad that the end is near. I would like to give thanks to the following people, without whom I would not have been able to finish this thesis.

First of all, I would like to thank my supervisor Dr. Zeki Erkin for his immense patience and supervision throughout my thesis. I appreciate all the feedback and guidance you have given me. Your cryptography courses have introduced me to the interesting world of privacy-preserving technologies, for which I am grateful. Further, I would like to thank my committee members, Prof. Inald Lagendijk and Dr. Stefanie Roos for taking time to read my thesis report and for being in my committee. I would like to thank everyone that has taken out time to read my thesis and providing me with valuable feedback. Thank you to the kind students who I got to know during my studies. You have made my studies more enjoyable. Finally, I want to give thanks to my family and friends, who have managed to put up with all my bullshit and have supported me unconditionally for all these years.

This thesis is dedicated to my mom and to the memory of my dad.

Seu Man To
Delft, November 2020

Contents

List of Abbreviations

BSN	'burgerservicenummer'
CP	cloud provider
CP-ABE	Ciphertext-Policy Attribute-Based Encryption
Dist-PACMAN	Distributed Polymorphic Access Management
Dist-PEP	Distributed Polymorphic Encryption and Pseudonymisation
EPD	'Elektronisch Patiëntendossier'
FHE	Fully Homomorphic Encryption
GP	General Practitioner
HCP	health care provider
IQS	Increasing Quorum Size
KS	keyserver
LSP	'Landelijk Schakelpunt'
OS	operating system
PACMAN	Polymorphic Access Management
PEP	Polymorphic Encryption and Pseudonymisation
PGO	'persoonlijke gezondheidsomgeving'
PHR	personal health record
SGX	Software Guard eXtension
SMDs	self-monitoring devices
TEE	Trusted Execution Environments
TEE-PACMAN	Trusted Execution Environment-based Polymorphic Access Management
TEE-PEP	Trusted Execution Environment-based Polymorphic Encryption and Pseudonymisation
TR	transcryptor
TRE	Trustworthy Remote Entity
UZI	'Unieke Zorgverlener Identificatie'
VZVZ	'Vereniging van Zorgaanbieders voor Zorgcommunicatie'

1 Introduction

The rise of electronic wearable devices has been tremendous over the past few years. Sales revenue is expected to reach $ 73 billion by 2022, starting from $ 16 billion in 2016 [49], with Bluetooth headsets, fitness bands and smartwatches being the most successful. The units of sold smartwatches have seen an increase from 0.6 million in 2013 to 15.3 million in 2018 in the U.S. alone [47], while the worldwide unit shipments have been forecasted to reach 109.2 million in 2023, coming from 69.3 million in 2019 [48].

Smartwatches are able to use sensors to monitor and track fitness and health-related data, such as heart rate, heart rate variability, respiration rate, steps taken, calories burned, sleep quality, and more. Since these devices are able to track medical measurements, the line between fitness and health trackers and medical devices have started to blur [14, 44]. One such device is the Apple Watch [1]. Besides tracking of basic metrics such as daily steps taken, it has the ability to identify an irregular heartbeat [19, 50], which could indicate atrial fibrillation. This is associated with an increased risk for a stroke, heart failure, and other heart-related complications [26]. Upon registering such an irregular heartbeat, the user gets a notification with the suggestion to contact a doctor. Another feature is the ability to make and record an electrocardiogram (ECG) using the user's smartphone [50]. From there, a report can be generated to send to the user's health care provider (HCP). Other more medical specialized self-monitoring devices (SMDs) are for example blood pressure monitors, glucose monitors and ECG monitors.

With the ongoing COVID-19 pandemic, SMDs have become a potential measure against the disease as they are constantly monitoring and collecting the user's vital signs, and they are widely available. Researchers have found that such wearable devices can be useful in detecting early warning signs of COVID-19 [35, 36]. The devices were able to identify some users with the disease before the onset of symptoms. If the accuracy of the detection can be improved, this could be a great tool in reducing the spread of the disease. Users would get a notification when the wearable suspects potential onset of COVID-19, generating more awareness for the user so that testing, quarantine and medical care can be executed at a more rapid pace.

Most SMDs upload the collected data to a cloud server through the user's smartphone for analytics and improved presentation of the data [44]. The increased usage of these devices makes them useful in the healthcare industry if the information collected by the devices is shared with HCPs [15, 30, 31]. The gathered information could then assist in identifying potential health issues, allowing prevention measures to be taken as opposed to treating diseases. Since patients are able to measure their medical condition continuously, remote monitoring is possible and could aid in avoiding emergency situations [5]. In addition, remote monitoring could lead to reduced visits to the doctor, which leads to reduced costs. Furthermore, collecting data from multiple patients in this manner could be a great source of data for research institutions.

Sharing medical data collected by SMDs with health care institutions has many benefits. However, medical data is sensitive and there are many concerns regarding its security and privacy when the data is stored on a cloud server. The cloud server, usually owned by the manufacturer of the device, should not be trusted with it, as consumer data and medical data have different legislation. Since the data gathered by the SMDs is considered consumer data, the manufacturer is able to exploit this data. However, the General Data Protection Regulation (GDPR) [2] is in effect since 25th May 2018, which introduced stricter rules regarding the handling of consumer's data. One solution to increase the privacy and security of this data, is to encrypt it

[1] `https://www.apple.com/nl/watch/`

before it gets stored on the cloud server. This way, the data remains private in case of a data breach. However, this leads to challenges in ensuring anonymity and providing proper access control.

1.1. Electronic healthcare in The Netherlands

In 2008 a bill has been submitted in The Netherlands to realize a country-wide electronic healthcare record, the 'Elektronisch Patiëntendossier' (EPD) [1]. The motivation behind this government initiated project is to facilitate sharing of medical data between health care providers, which would lead to the reduction of medical failures due to misinformation.

However, due to privacy and security concerns, the government pulled out of this project in 2011. This eventually led to the formation of a union of HCPs, the 'Vereniging van Zorgaanbieders voor Zorgcommunicatie' (VZVZ) [2], which would eventually be responsible for creating, managing, and the further development of the 'Landelijk Schakelpunt' (LSP) [3]. The LSP is an infrastructure specifically created for the exchange of medical data between HCPs.

It is mandatory for every HCP to keep medical health records of their patients [3, 34]. Almost all HCPs keep these records electronically, and thus have their own local EPD. To exchange data from one HCP to another, the patient has to explicitly give consent for their information to be shared through the LSP. This can be achieved through filling out a consent form in person, online, or verbally. When consent is given, the patient's 'burgerservicenummer' (BSN), which is the national identification number, gets registered at the LSP. HCPs can access the LSP with a UZI-card and a PIN code. 'Unieke Zorgverlener Identificatie' (UZI) is a unique identification number for HCPs. Whenever a HCP wants to search for the patient's medical data, it looks up the patient's BSN through the LSP to see where the data is located. Thus no actual medical data from the patient is stored at the LSP. The General Practitioner (GP) only shares the patient's Professional Summary and the pharmacy only shares the medication record. No other information is shared through the LSP. Additionally, the HCP requesting this information is only allowed access if the patient has given consent to the HCP and the information is needed for the diagnosis and/or treatment of the patient.

Currently, the LSP is divided into 44 regions. Some regions are managed by their own region manager or by regional organisations, while some regions are managed by VZVZ. However, the regions managed by a region manager or regional organisations are required to update VZVZ about changes inside their regions. HCPs can only share and access medical data within their region. If an HCP is located near multiple regions or is serving patients from other regions, they are allowed to register at more regions. However, hospitals are allowed to register at multiple regions as their patients come from different parts of the country, in contrast to the local GP or pharmacy. Due to COVID-19, this "region lock" has been lifted temporarily [52]. Due to capacity issues, some patients cannot get treated in their own region and therefore have to receive hospital care in other regions. All urgent care and COVID-19 testing facilities are able to request the patient's medical data through the LSP.

Patients have access to an online platform 'Volg je zorg' [4], from where they can invoke and revoke the permission from the HCPs to access their data. Further, they can observe when and which HCP has accessed their medical health record, and the kind of data that has been shared. However, patients cannot access their own medical health records. If they wish to see the actual content, they can request this at each HCP individually. In addition, starting from 1st July 2020, patients are entitled to electronically inspect their own medical health record [4]. The HCP has to provide the patient their record. This can be accomplished by sending the record through e-mail, getting the patient access through an online portal from the HCP, or through a 'persoonlijke gezondheidsomgeving' (PGO).

A PGO is an application in which the patient has access to their own personal health record (PHR) [39]. The patient can also add data to this record, such as their blood group, medications and vaccinations. PGOs are still under development and are currently being tested. The goal of such an application is to eventually let the patient take control over their own medical data. Ultimately, patients should be able to gather the medical data from their HCPs and their pharmacy in one place. They should be able to share their data with other HCPs, and should be able to add data to their own record whether by hand or by uploading from other applications or wearables.

However, a PGO is not a replacement for the LSP. In a PGO the patient can inspect their medical data, while this is not possible with the LSP. The LSP's main contribution is facilitating the exchange of information

[2] https://www.vzvz.nl/
[3] https://www.vzvz.nl/over-het-lsp
[4] https://www.volgjezorg.nl/

between HCPs and pharmacies 24/7. In case of an emergency, the first aid responders request the patient's medical data through the LSP.

1.2. Polymorphic Encryption and Pseudonymisation

Polymorphic Encryption and Pseudonymisation (PEP) [51] is a framework that facilitates the exchange of medical data between patient, self-monitoring devices and health care providers while providing proper access management and security. PEP allows the data to be stored at a cloud provider (CP) in encrypted form, while guaranteeing that the CP does not learn anything about the contents and owners of the data. Anonymity is attained through pseudonymisation, which means that different identifiers are used at different parties for the same data. Each party owns a secret key that is related to the master secret key $\mathcal{X}$ in such a way that re-keying of encryptions can be performed without knowing $\mathcal{X}$. The keyserver (KS) is the only party that knows $\mathcal{X}$.

The key party in this protocol is the transcryptor (TR). The TR is a semi-trusted intermediate party that performs the re-keying and re-shuffling of encryptions and where all messages go through. Re-keying of an encryption is the act of changing an encryption so that another secret key can be used to decrypt it. Re-shuffling of an encrypted identifier is the act of changing the encryption in such a way that when it is decrypted, the identifier looks different from when it got encrypted. Since the TR performs these actions on data in encrypted form, it does not learn anything about the content. However, it does register who is sharing data with whom, enabling access management and logging.

Both the KS and the TR handle secret keys that are critical to the security and privacy of the system. If the KS and the TR collude, they are able to derive the secret keys from all the participants, and are able to decrypt all data and match it to the patients. Furthermore, if the TR colludes with any participant, the master secret key can be obtained, making it possible to decrypt all data. Aside from the issue of collusion, scalability becomes an issue with more participants joining the system. In addition, the whole system stops working if the TR goes offline.

1.3. Distributed-PEP and Polymorphic Access Management

Distributed Polymorphic Encryption and Pseudonymisation (Dist-PEP) [32] [33] is a solution to some of the problems of PEP. In Dist-PEP, the role of the TR is distributed over a group of n TRs, of which a quorum π of t TRs need to collaborate in order to perform the re-keying and re-shuffling operations. Any group of $t-1$ or less TRs is unable to perform these operations or learn secret keys, even with the collaboration of HCPs and/or the CP. This is achieved by using Shamir Secret Sharing [45]. Further, the TRs take over the role of KS by generating the master secret key and the user secret keys through secret sharing of random values. Every TR owns only a share of the keys. Dist-PEP facilitates scalability and availability, while improving security over PEP.

Polymorphic Access Management (PACMAN) [32] removes the need for an access manager by embedding the access structure into the encryption. This is achieved by using Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [8]. Only a party holding all the attributes to satisfy the access structure under which the data is encrypted, is able to decrypt. The leafs from the access structures are pseudonymised, ensuring the attributes from being leaked.

Combining Dist-PEP and PACMAN is more robust than PEP as it facilitates availability and scalability, while improving security by lowering the probability of successful collusions.

- Availabilty: provided that at least t TRs stay online, the system remains operational.
- Scalability: when more parties join the system, the introduction of more TRs to the system is possible.

In addition, the owner of the data is given the power to decide who gets access to which parts of his data. Access management is turned into the distribution of the correct attributes to each party.

Although Dist-PEP and PACMAN are an improvement to PEP, the TRs still hold a lot of power as all the computations are performed by them: key generation, re-keying, and replacement of all identifiers with pseudonyms. This requires them to collaborate on all the computations, which is computationally expensive. Moreover, as more TRs get added to the system for more stability when the number of participants grows, it increases the risk of reaching the threshold of a quorum of malicious TRs, allowing them to collaborate successfully.

1.4. Trusted Execution Environments

There have been many works that use homomorphic encryption in the processing of sensitive data in the medical domain [9, 11, 25, 27, 28]. With the introduction of Trusted Execution Environments (TEE) in processors, like ARM's TrustZone [5] and Intel's Software Guard eXtension (SGX) [6], new solutions for processing medical data securely are offered. These processors can be found in high-end servers to personal computers and mobile devices. A TEE is a tamper-resistant isolated area of a processor that guarantees integrity and confidentiality of the executed code and data. It protects code and data against various attacks, including the operating system (OS). When data leaves the TEE, it is encrypted and integrity protected.

Since SMDs are generally able to connect to smartphones and laptops through Bluetooth or Wi-Fi to upload the data gathered to a cloud provider, using TEEs in these devices could possibly alleviate the TRs from some responsibilities, while preserving privacy and security.

1.5. Research objective

The widely adaptation of self-monitoring devices has the potential to improve electronic healthcare. Integrating these devices into the Dutch e-health could lead to improved diagnosis and treatment. The current Dutch e-health does not support such an integration, although there are ongoing developments towards a similar solution. However, this solution is comparable to a personal health record for the patient. Health care providers are not able to access this data in case of emergencies.

The main goal of this thesis is to make improvements to the efficiency and scalability of the PEP framework combined with the Dist-PEP and PACMAN extensions, while retaining privacy and security for use in the Dutch electronic healthcare landscape. The research question is formulated as follows:

How can we use integrated TEEs in SMD-connected devices to reduce the computation load of the TRs and make the PEP framework more efficient and scalable, while preserving privacy and security?

To aid in answering the research question, the following subquestions are defined:

1. What other works related to medical data privacy and security are there and how do they make use of trusted execution environments, if any?

2. What (cryptographic) methods are there to ensure a secure and privacy-preserving electronic personal health record?

1.6. Contribution

The main contributions of this thesis are:

- Presenting a protocol for increasing the size of a quorum of TRs from t to t', where $t' > t$, after set-up, lowering the chance of a quorum of t' malicious TRs to collaborate.

- Presenting a protocol that uses trusted execution environments in SMD-connected devices to take over the re-keying and replacement of identifiers by pseudonyms in Dist-PEP and PACMAN, effectively reducing the computation load and responsibilities from the TRs, while preserving privacy and security. Moreover, increasing the scalability as every patient owns an SMD that performs most of the computations instead of a group of TRs.

These contributions lead to an electronic personal health record, managed by the patient, where multiple parties are able to exchange medical data of the patient in a secure and privacy-preserving manner. The patient decides what type of data, and who is able to retrieve and store data to its personal health record.

1.7. Thesis outline

The rest of this thesis is structured as follows. In Chapter 2, we explore what other research exists regarding trusted execution environments in electronic healthcare. Chapter 3 discusses (cryptographic) preliminaries that are required to be understood for the rest of this thesis, followed by Chapter 4 in which we present and

[5] `https://developer.arm.com/ip-products/security-ip/trustzone`
[6] `https://www.intel.com/content/www/us/en/architecture-and-technology/software-guard-extensions.html`

implement a protocol for increasing the size of the quorum of TRs after set-up and provide an analysis of its complexity and performance. Chapter 5 introduces our Trusted Execution Environment-based Polymorphic Encryption and Pseudonymisation (TEE-PEP) and Trusted Execution Environment-based Polymorphic Access Management (TEE-PACMAN) protocols including a security and complexity analysis. Finally, we conclude by providing a discussion and describe future work in Chapter 6.

2

Related Works

In this chapter, we investigate other existing works on secure sensitive data storage and processing. Since privacy-preserving data managing is a large field with applications in several domains, we solely focus on works related to processing medical data and on solutions using TEEs.

2.1. KONFIDO

KONFIDO [46] is an OpenNCP-based Secure e-Health Data Exchange system. OpenNCP [22] is a platform that enables secure access to patient health information between different European healthcare systems, and it is a result of the epSOS (European Patients - Smart Open Services) project. The main goal of the epSOS project is to achieve interoperability between European healthcare systems while complying with both National and European laws. KONFIDO builds on and extends the OpenNCP framework to increase trust and security of e-health data exchange.

2.1.1. OpenNCP

OpenNCP is a practical e-health framework and ICT infrastructure, based on existing National Infrastructures, that enables secure access to patient health information among European healthcare systems. It provides two services: the epSOS' Patient Summary for EU Citizens and the epSOS' ePrescription/eDispensing for EU Citizens. Patient Summary is a set of basic medical data that includes the most important information required to ensure safe and secure healthcare, e.g. a patient's general information, clinical data, prescribed medicine, etc. This information is useful in case of an emergency or accident, and can be retrieved by health care professionals. ePrescription/eDispensing contains the currently available electronic prescription of medication, which is the medication that is dispensed in the patient's home country. This allows the patient to get the medication in different European countries if necessary. The following key aspects used in the epSOS project makes e-health interoperability in the EU possible:

- The existing national health care infrastructures and legislation remain the same.
- Trust among EU Member States is based on contracts and agreed policies.
- Information is exchanged but not shared.

OpenNPC uses a set of interacting National Contact Points (NCPs). The NCPs act as the connection between the National Infrastructures. They process and transfer medical data between countries. Figure 2.1 shows the OpenNPC high-level architecture. A NCP has 5 main components:

- Data Discovery/Exchange services: establishes communication and allows the identification of patients and the retrieval of medical data.
- Trust Services: ensures the circle-of-trust, i.e. it takes care of the validation, verification, signing and mapping of messages and information, and ensures the patient consent mechanism.
- Transformation Services: used to transform clinical documents (translation and mapping of taxonomy).

- Audit Services: manages the logging of the system and the traceability thereof and assures operation auditing.
- Support Services: ensures the availability of the service, such as response time, and guaranteed message delivery and session.

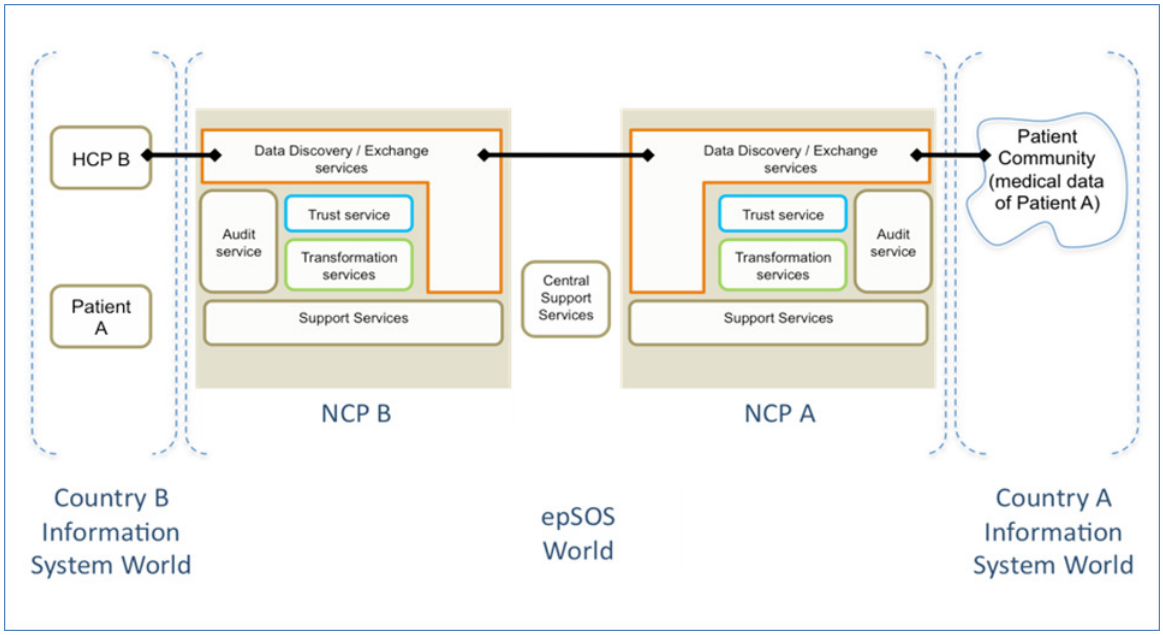

Figure 2.1: OpenNCP high-level architecture diagram [22].

The Circle of Trust is among the NCPs and they are able to establish mutual trust relationships. A NCP acts as a legal entity that creates a secure link between the epSOS trust domain from the national trust domain. Only the NCPs have an identity in both domains.

We describe a Patient Summary request scenario in figure 2.1. Patient A from country A attends a HCP B in country B. HCP B requests patient A's Patient Summary from the National Infrastructure of its country. The National Infrastructure makes contact with NCP B. Then NCP B interacts with other NCPs, in this case NCP A, and uses all the services they have to retrieve the information through country A's National Infrastructure in the right format.

The communication security from OpenNCP is ensured by cryptography and secure protocols. However, the security of the communicating parties is only enforced by a legally binding agreement. Therefore, if a National Infrastructure is compromised, it can exploit the NCP. Further, the medical data is in plain text in almost all the services offered by the NCPs. Thus, a compromised NCP can turn into a data breach.

2.1.2. KONFIDO toolbox

KONFIDO overcomes some of the vulnerabilities of OpenNCP by guaranteeing that medical data is never in plain text in non-secure areas. KONFIDO uses the following services and tools:

- Trusted Execution Environment (TEE): the TEE used in KONFIDO is created using Intel's SGX. Certain functions of OpenNCP are performed in the TEE. For example, during the exchange of the Patient Summary, the data is being decrypted, transformed and re-encrypted. During this time, the data is exposed. Thus, these functions will be performed inside the TEE. In addition, remote attestation is provided by SGX, which is a mechanism in which a host presents reliable evidence about its hardware and the software it is running to a remote host. Hence, the TEE from a NCP must convince the TEE from another NCP with whom it is communicating with, that it has a valid measurement hash, runs in a secure environment and that it has not been tampered with.
- Physical Unclonable Function (PUF): it uses a PUF-based random number generator for the generation of keys. The random numbers generated using the PUF are practically impossible to predict or manip-

ulate. These PUF-based random number generators are used in the NCP, and the random numbers are securely provided to the TEE for key generation.

- Homomorphic Encryption: allows computation on encrypted data. In this way, the medical data remains protected during transport, storage and processing.
- Security Information and Event Management (SIEM): the KONFIDO SIEM extends some existing SIEM solutions and customizes them. The KONFIDO SIEM is able to work with homomorphic encrypted data, communicate with TEEs, deal with federated deployment characteristics of OpenNCP-compliant scenarios, and provide encrypted output using a PUF-based encryption technique.
- Blockchain-based auditing system: provides an audit trail that a certain party has requested specific medical data and that the request has been accepted or denied. A block chain is a distributed unforgeable ledger. It is a chain of blocks linked through cryptographic hashes. Each block is linked to its predecessor.
- eIDAS-based authentication system: eIDAS stands for Electronic Identification, Authentication and Trust Services. It is a regulation on electronic identification and trust services for electronic transactions within the EU internal market. eIDAS-compliant authentication in KONFIDO exists in two forms: one for health care providers and one for patients. Health care providers must access the system with a strong digital identity which is issued by their country of residence. Patients can access the system using an eIDAS cross-border authentication.
- Publish/subscribe communication channel: each national infrastracture has a dedicated SIEM. Each SIEM is interconnected with other SIEMs in order to exchange security metrics through a secure publish/subscribe communication channel.
- TEE communication channel: trusted communication channels are established through remote attestation of TEEs.

Figure 2.2 shows the Patient Summary response with KONFIDO. The actions inside the `opt` rectangles are performed in a TEE and are supported by other KONFIDO services.

Figure 2.3 depicts the KONFIDO architecture. The following services are deployed in each NCP:

- The actions performed to transform the Patient Summary happen inside a TEE.
- A PUF-component is used to generate keys and certificates, and to secure communication channels.
- eIDAS is used to improve authentication.
- The block chain based auditing system is used for log management and storing.
- Homomorphic encryption techniques are used for processing data in encrypted form.

The following services are deployed in each National Infrastructure:

- To secure the transmission of a Patient Summary, a TEE is required.
- All other services are optional.

KONFIDO offers different kinds of communication channels to deal with the vast amount of different devices that are involved in OpenNCP.

- TEE communication channel: allows the exchange of data between SGX based TEEs.
- Secure Sockets Layer (SSL) communication channel between SGX-based TEE and other TEEs: allows the exchange of data between TEEs based on different technologies, e.g. Intel SGX and ARM TrustZone.
- Homomorphic Encryption + SSL communication channel: this channel is used when TEE technologies are not available in some devices.
- SSL communication channel: used for local data exchanges only, e.g. communication between the PUF hardware and the TEE.
- OpenNCP communication channel: this is the standard OpenNCP communication channel.

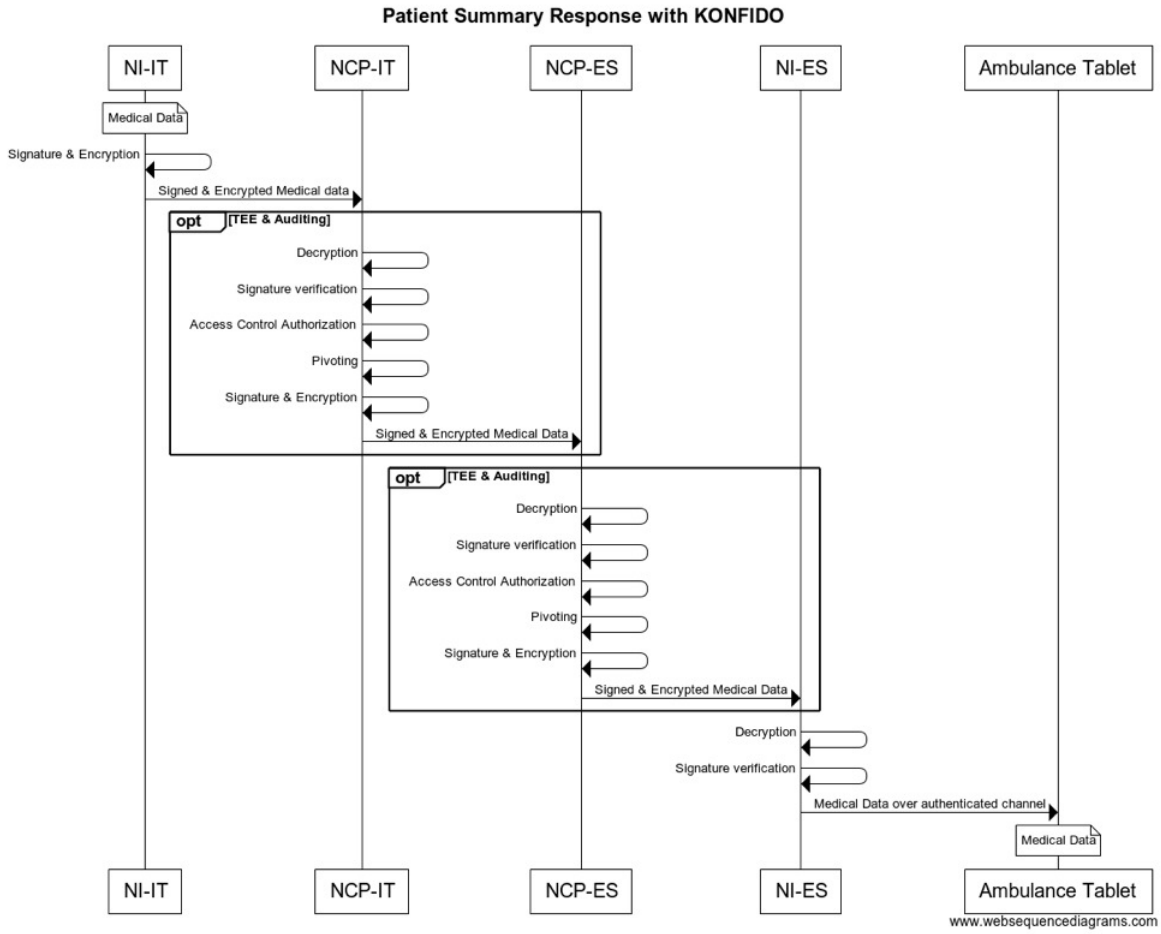

Figure 2.2: Patient Summary Response with KONFIDO [46].

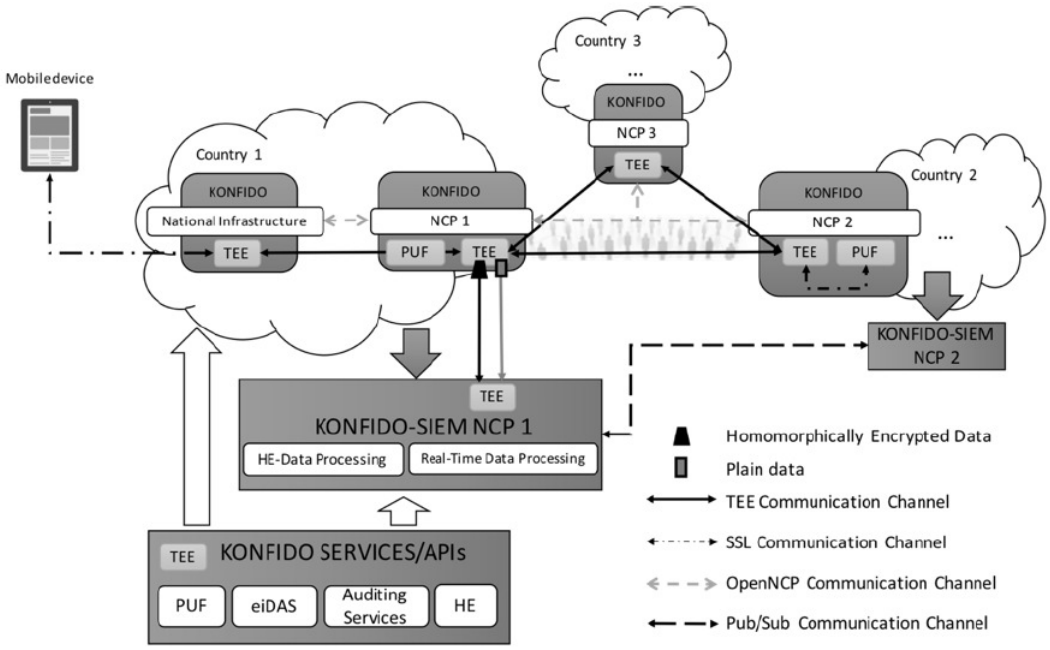

Figure 2.3: KONFIDO Architecture [46].

2.2. SGX-SPARK

SGX-SPARK [43] is a system for processing data streams generated by health-monitoring wearable devices. It uses Spark Streaming [54], which is an extension of Spark [53], for fault-tolerant stream processing of data streams and the processing takes place inside TEEs in untrusted clouds. Intel's SGX is used for the implementation of the TEE. In SGX, the trusted execution environment is called an enclave. The scenario taken for designing the system is where multiple sensors track the cardiac activity of different users, which is used to study the Heart Rate Variability (HRV). HRV is the variation in the time intervals between consecutive heartbeats and can be a predictor for myocardial infarction. The system is designed for heart-specific signals, but it can be changed so that it can be used for different stream processing systems. Figure 2.4 depicts the architecture of SGX-SPARK. The server executes on untrusted clouds with SGX, and the clients are the sensors that generate the data. The interaction between them happens through a filesystem interface. On the server, SGX-SPARK processes the data using different HRV analysis algorithms.

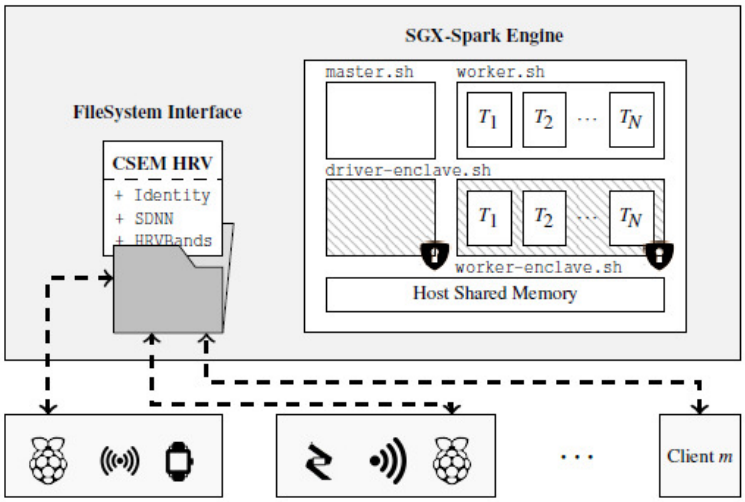

Figure 2.4: SGX-SPARK system architecture [43].

The performance of SGX-SPARK is compared to Vanilla Spark, which is the non-secure version of Spark. SGX-SPARK has a performance overhead of 4 to 5 times in comparison to Vanilla Spark. However, for typical signal processing, it is still practical.

2.3. Privacy-preserving medical data analytics solutions using fully homomorphic encryption

In [27] and [28], Fully Homomorphic Encryption (FHE) is used for computations on encrypted health information. FHE schemes allow both addition and multiplication operations on ciphertexts, in contrast to partially homomorphic schemes, which allow only addition or multiplication operations. The authors have implemented a long-term cardiac health monitoring application that allows the cloud, where the data is stored in encrypted form, to perform computations on it. The system is depicted in figure 2.5. The FHE scheme used is the Brakerski-Gentry-Vaikuntanathan (BGV) [13] scheme. Despite the application's performance disadvantage, it can be deployed by healthcare organizations.

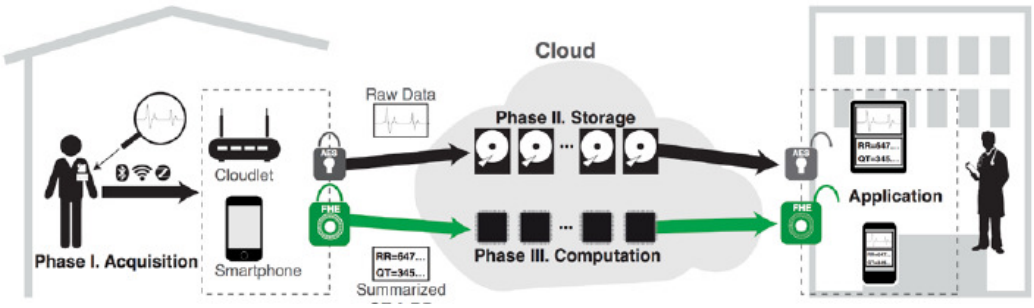

Figure 2.5: Cloud-based long term health monitoring system setting [27].

Another work that uses FHE for computations on encrypted personal health data is SafeBioMetrics [9]. Its application domain is the secure storage and processing of personal health and physiological information obtained through mobile sensing for e-health applications. Figure 2.6 shows the system architecture. As can be observed, FHE is used for the computation on the encrypted data obtained by the mobile client device (data processing path), while the standard encryption scheme AES (Advanced Encryption Standard)is used for normal secure storage and retrieval of the data without computations (raw data path). The FHE scheme used is the Brakerski-Gentry-Vaikuntanathan (BGV) [13] scheme. The specific use of the SafeBioMetrics systems is gathering cardiac rate data and performing computations on it to detect abnormalities. In addition, it can be modified to accommodate other scenarios concerning data collection through mobile devices.

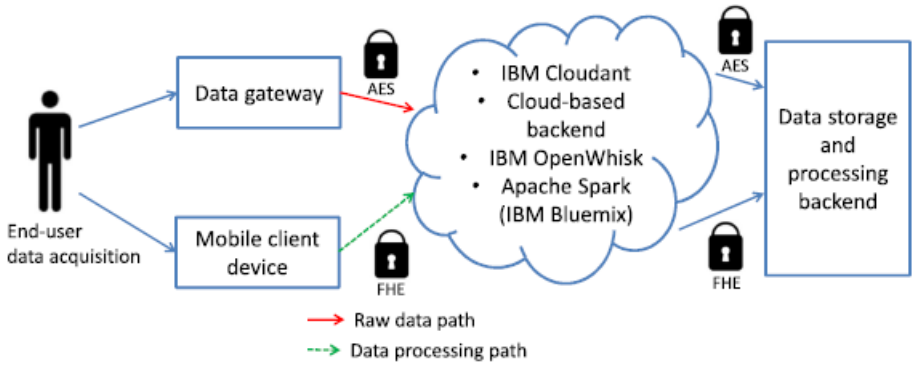

Figure 2.6: SafeBioMetrics system architecture [9].

In [11], a cloud service has been implemented that can compute the likelihood of a person suffering from a heart attack based on certain body measurements. The cloud service computes on encrypted data using homomorphic encryption, so that nothing can be learned about the actual content. The service works as follows. The user uses a device, e.g. a personal computer, with the client application installed that collects the user's health data, encrypts the collected data, and sends it to the cloud application. The cloud application runs an algorithm over this data to make a prediction, and sends back an encrypted prediction to the client application. Then the user can decrypt to become aware of the probability of having a heart attack. The used homomorphic scheme is the more practical variant of the Ring-Based Fully Homomorphic Encryption scheme in [12].

More practical FHE schemes remain an active research area.

2.4. Privacy-preserving data analytics solutions using trusted execution environments

In [24], the authors describe and implement a system where an Internet of Things (IoT) middleware is run inside a TEE using Intel's SGX. Processed data is being encrypted inside the SGX enclave so that it is unreadable for the system hypervisor. This facilitates the deployment of IoT middleware on untrusted platforms. Figure 2.7 depicts the architecture of the system. The application logic is performed inside the enclave. The enclave communicates through a webserver with the users and the IoT devices.

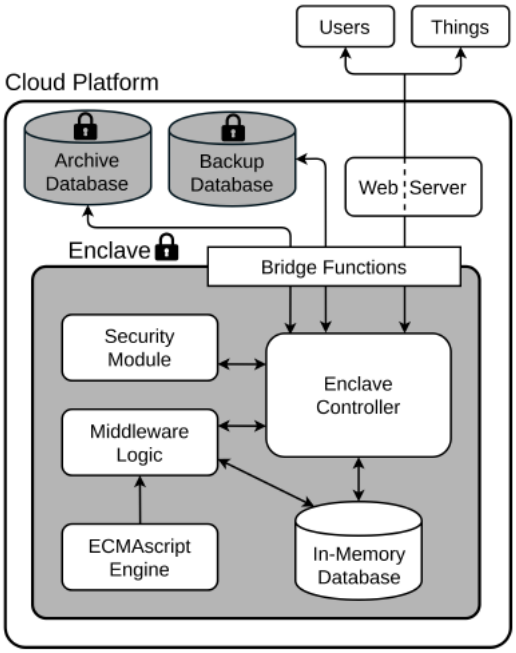

Figure 2.7: System architecture [24].

Another work that uses a TEE is [29]. Intel's SGX is used to implement a Trustworthy Remote Entity (TRE) [41] [40] for use in multi-party applications. A TRE is in essence a Trusted Third Party (TTP) that provides a high level of assurance of its state and behaviour, making it trustworthy instead of trusted. In many-party applications, the TRE acts as the intermediate party in the communication between participants and performs privacy-enhancing operations on the communicated data. In this work, the multi-party application is privacy-preserving energy metering. Figure 2.8 shows the system overview of a TRE in a smart grid. The arrows show the sequence of communication between the Service Provider (SP) and the Smart Measuring Devices (SMDs). The SP and SMDs do not trust each other. The SP sends requests to the SMDs through the TRE, and the SMDs respond through the TRE. While TRE is fetching the data from the SMDs, it can perform privacy-preserving operations on the data before sending it to the SP.

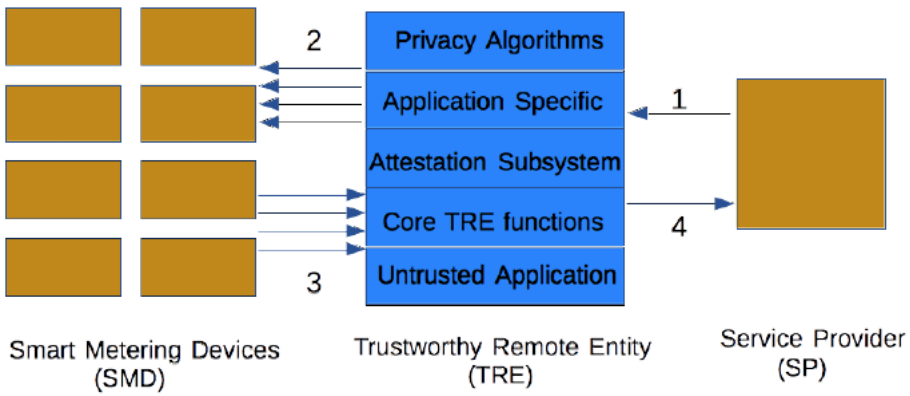

Figure 2.8: System overview of a TRE in a smart grid [29].

In [17], client-side disk/file encryption is performed using Intel's SGX. The Cryptomator, which is an open source client-side encryption tool used for protection of files stored on clouds, is integrated with Intel's SGX technology. The data are encrypted and decrypted inside the SGX enclave. The enclave can seal the data by using a sealing key. The sealing key is a unique key generated by the CPU for this particular enclave on this particular platform. Therefore, the data can only get unsealed inside the enclave. After sealing the data, it can be written to the storage device that synchronizes with the cloud storage service.

In [20], the authors combine homomorphic encryption with a TEE to guarantee code and data integrity and privacy. The combined model is useful for remote servers handling data and computations. Intel's SGX and Paillier encryption scheme [38] are used for the implementation as respectively the TEE and the homomorphic encryption scheme. The combined model considers four entities:

1. A user, the owner of the private data.
2. An enclave that the user can audit and assess. If the user receives an attestation that the enclave is securely loaded on the remote platform, then the user can trust the enclave.
3. An untrusted application that launches the enclave and connects it to the OS of the server.
4. A database holding private information.

Figure 2.9 shows how the user stores data at the database. The user (Client) and the enclave first establish a secure channel. Subsequently, the user encrypts the data with Paillier encryption and sends it to the enclave. The enclave re-encrypts the data and sends it to the database for storage.

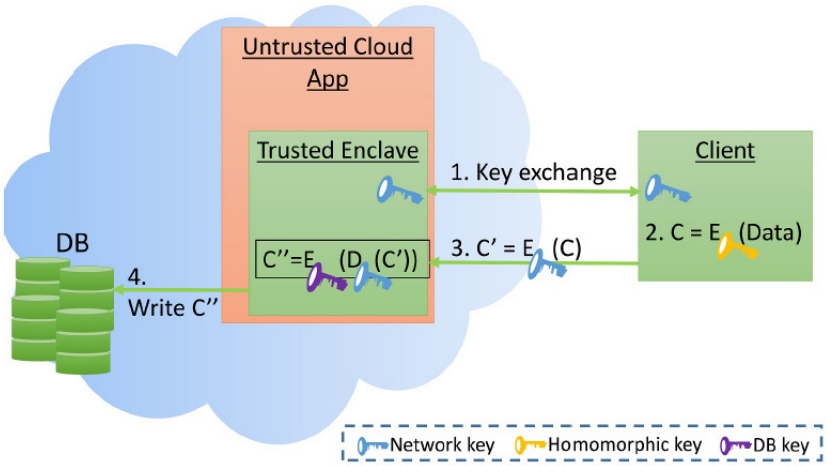

Figure 2.9: Storing data in the database [20].

Figure 2.10 shows how the user retrieves data from the database. The user (Client) and the enclave first establish a secure channel. After, the user sends a query request to the enclave. The enclave retrieves the encrypted data from the database and performs the required computations on the encrypted data. Then it re-encrypts the result and sends it to the user. Finally, the user decrypts the encryption to obtain the data.

The performance of a system that only uses partial homomorphic encryption and the performance of a system that only uses TEE are roughly the same, with the performance of using partial homomorphic encryption being slightly better. The performance of a system that uses both partial homomorphic encryption and TEE (i.e. the combined model) is 1.7 times slower.

2.5. Conclusion

Many existing works on secure private data storage and/or processing use either homomorphic encryption, trusted execution environments or a combination of both. The TEEs are mostly integrated on the cloud servers for that is where the data is stored and processed most of the time. By using TEEs on the cloud servers, it is guaranteed that the cloud servers cannot observe the data. Additionally, using homomorphic encryption allows the cloud servers to perform computations on encrypted data without learning anything about the contents. Furthermore, using both a TEE and an encryption scheme provides an extra layer of protection. If the encryption scheme is broken successfully, the data is protected by the TEE. Conversely, if the TEE is successfully attacked, the data is protected by the encryption scheme.

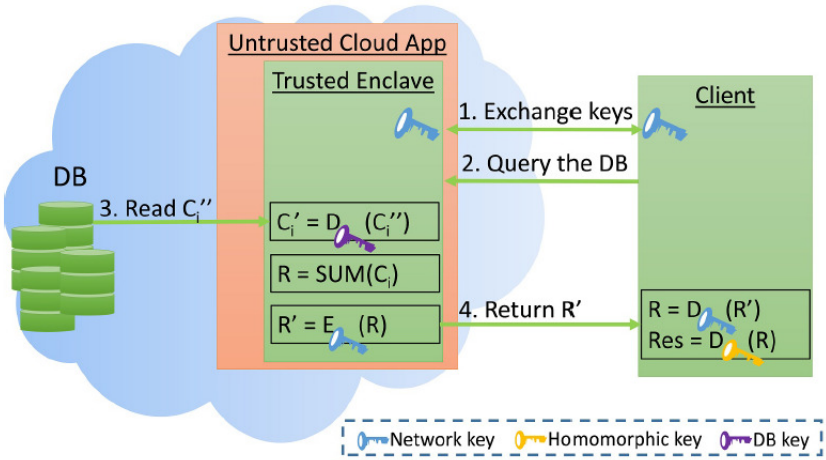

Figure 2.10: Retrieving data from the database [20].

3

Preliminaries

This thesis and its prior work are built upon existing (cryptographic) schemes and protocols. In this chapter, we introduce ElGamal encryption, Shamir Secret Sharing, Ciphertext-Policy Attribute-Based Encryption (CP-ABE), Trusted Execution Environments (TEE), two variants of the Polymorphic Encryption and Pseudonymisation (PEP) protocol and two variants of the Polymorphic Access Management (PACMAN) protocol.

3.1. ElGamal encryption

ElGamal encryption is an asymmetric, public-key cryptosystem and its security depends on the difficulty of computing discrete logarithms over finite fields [21].

3.1.1. Description

Key Generation. Let p and q be large primes such that $p = 2q+1$ and let $g \in \mathbb{F}_p^*$ be a generator of multiplicative subgroup $\mathbb{G}_q$ of order q. The private key $\mathcal{X}$ is a random value $x < q$. The corresponding public key is $\mathcal{Y} = g^{\mathcal{X}} \mod p$.

Encryption. Encryption uses a random value $r < q$. Encrypting a message M under public key $\mathcal{Y}$ is performed as

$$\mathrm{EncEG}_{\mathcal{Y}}(M) = \langle g^r, \mathcal{Y}^r \cdot M \rangle \mod p. \tag{3.1}$$

Decryption. Decryption of a ciphertext of the form $\langle b, c \rangle$ with private key $\mathcal{X}$ is performed as

$$\mathrm{DecEG}_{\mathcal{X}}(\langle b, c \rangle) = \frac{c}{b^{\mathcal{X}}} = \frac{\mathcal{Y}^r \cdot M}{g^{r\mathcal{X}}} = \frac{g^{\mathcal{X}r} \cdot M}{g^{r\mathcal{X}}} = M \mod p. \tag{3.2}$$

3.1.2. Multiplicative homomorphic

Homomorphism is a property that an encryption scheme can possess that allows performing certain computation on the ciphertext without decrypting it. The ElGamal encryption scheme has the multiplicative homomorphism property, which means that given the encryptions of two messages m_1 and m_2, the encryption of $m_1 \cdot m_2$ can be determined without knowing the contents of m_1 and m_2:

$$\begin{aligned} \mathrm{EncEG}_{\mathcal{Y}}(m_1) \times \mathrm{EncEG}_{\mathcal{Y}}(m_2) &= (g^{r_1}, \mathcal{Y}^{r_1} \cdot m_1)(g^{r_2}, \mathcal{Y}^{r_2} \cdot m_2) \\ &= (g^{r_1+r_2}, \mathcal{Y}^{r_1+r_2}(m_1 \cdot m_2)) \\ &= \mathrm{EncEG}_{\mathcal{Y}}(m_1 \times m_2). \end{aligned} \tag{3.3}$$

3.2. Shamir Secret Sharing

Shamir Secret Sharing [45] is a (t, n) threshold secret sharing scheme. A secret S is divided into n shares and at least $t \leq n$ shares are needed in order to retrieve S.

Let S be the secret to be divided into n shares and f be a polynomial of degree $t-1$ with $f(0)=S$. Each party i, with $i=\{1,...,n\}$, receives a unique point $(x_i,y_i)=(i,f(i))$ on f, where x_i is public and y_i is the secret share. Given t unique points on f, f can be recovered using Lagrange interpolation and hence the secret S. Secret S can be recovered by computing $S=f(0)$ by using

$$f(x)=\sum_{i\in\pi} y_i \ell_i(x) \mod p, \tag{3.4}$$

where

$$\ell_i(x)=\prod_{j\in\pi, j\neq i} \frac{x-x_j}{x_i-x_j} \mod p. \tag{3.5}$$

3.2.1. Addition of secrets

Let $[S_1]$ and $[S_2]$ be two secret sharings with respectively polynomials f and g, both with degree $t-1$. A secret sharing $[T]=[S_1+S_2]$ [7, 10] can then be generated as follows.

Assume that $[S_1]$ and $[S_2]$ are shared among n parties and every party i holds the shares $f(i)$ and $g(i)$. Every party i computes $h(i)=f(i)+g(i)$ to obtain its secret share for $[T]$.

3.2.2. Multiplication of secrets

Let $[S_1]$ and $[S_2]$ be two secret sharings with polynomials f with degree t_f-1 and g with degree t_g-1, respectively. A secret sharing $[T]=[S_1\cdot S_2]$ can then be generated as follows [7, 10].

Let π be a group of parties, with $|\pi|=t_f+t_g-2$. Every party $i\in\pi$ holds the shares $f(i)$ and $g(i)$, and computes $h(i)=f(i)\cdot g(i)$. The degree of h is then t_f+t_g-2. To obtain a secret sharing with threshold $t_h\leq t_f+t_g$, every $i\in\pi$ generates a secret sharing $h'_i=h(i)\prod_{j\in\pi,j\neq i} j$, and gives every other party j a share $h'_i(j)$. After, every party i computes their secret share of h'or$[T]$ as $\sum_{j\in\pi} h'_j(i)$.

3.2.3. Exponentiation with a secret

Consider a secret sharing $[S]$ and a value v. v^S can then be computed without revealing the secret S [18]. Given a quorum π with t parties, every party $i\in\pi$ calculates $S_i=y_i\ell_i(0)$ using equation 3.5, and $p_i=v^{S_i}$ as partial result. Finally, the partial results are combined to obtain $\prod_{i\in\pi} p_i=\prod_{i\in\pi} v^{S_i}=v^S$.

3.2.4. Generation of a random secret

To generate a secret sharing $[S]$ of a random value with threshold t [6], every party i chooses a random polynomial r_i of degree $t-1$. Then they calculate for every party j a share $r_{i,j}$ and sends it to them. After, every party j computes $\sum_{i=1}^{n} r_{i,j}$ to obtain their secret share of $[S]$.

3.2.5. Inverse of a secret

Let $[S]$ be a secret sharing with polynomial f. The inverse of $[S]$, $[S^{-1}]$ can be computed [6] as follows. First, a secret sharing of a random value $[R]$ is generated, then secret shares for $[I=SR]$ are computed. Afterwards, the value of I is revealed and $[I\cdot S]=[(SR)^{-1}R]=[S^{-1}]$ is computed.

3.3. Ciphertext-Policy Attribute-Based Encryption

In Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [8], data is encrypted according to an access policy over attributes. This access policy can be defined as an access tree with the attributes at the leaf nodes and threshold values at the internal nodes. The access tree is embedded in the encryption, while the decryption key is associated with a set of attributes. The ciphertext can only be decrypted by a party holding the decryption key with the right attributes to satisfy the access tree.

3.3.1. Bilinear maps

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two mulitplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}_0$ and e be a bilinear map, $e:\mathbb{G}_0\times\mathbb{G}_0\to\mathbb{G}_1$ with the following properties:

1. Bilinearity: for all $u,v\in\mathbb{G}_0$ and $a,b\in\mathbb{Z}_p$, we have $e(u^a,v^b)=a(u,v)^{ab}$.
2. Non-degeneracy: $e(g,g)\neq 1$.

If the group operation in $\mathbb{G}_0$ and the bilinear map e are efficiently computable, then $\mathbb{G}_0$ is considered a bilinear group. In addition, note that e is symmetric: $e(g^a, g^b) = e(g,g)^{ab} = e(g^b, g^a)$.

3.3.2. Access tree

Let $\mathcal{T}$ be a tree depicting an access structure. Each internal node x represents a threshold gate, and holds a threshold value t_x and its set of children S_x. Every leaf node x holds a threshold value $t_x = 1$ and an associated attribute att_x. If num_x is the number of children of a node x, then $0 < t_x \leq num_x$. When $t_x = 1$, the threshold gate is an OR gate and when $t_x = num_x$, it is an AND gate. Let $parent(x)$ be the parent of node x, and $index(x)$ be the index of x within $S_{parent(x)}$, which ranges between 1 and $|S_{parent(x)}|$.

3.3.3. Satisfying an access tree

Let $\mathcal{T}$ be an access tree with root r and γ be a set of attributes. Let $\mathcal{T}_x$ denote the subtree of $\mathcal{T}$ rooted at x. Thus $\mathcal{T}$ is $\mathcal{T}_r$. If γ satisfies access tree $\mathcal{T}_x$, then this is denoted as $\mathcal{T}_x(\gamma) = 1$. $\mathcal{T}_x(\gamma)$ is recursively computed as follows. If x is a leaf node and $att_x \in \gamma$, then $\mathcal{T}_x(\gamma) = 1$. Otherwise x is an internal node, and $\mathcal{T}_{x'}(\gamma)$ is being computed for all children x' of x, $\mathcal{T}_x(\gamma) = 1$ if $\sum_{x' \in S_x} \mathcal{T}_{x'}(\gamma) \geq t_x$.

3.3.4. Construction

Setup. Let $\mathbb{G}_0$ be a bilinear group of prime order p with generator g and bilinear map e. Choose $\alpha, \beta \in_R \mathbb{Z}_p$ and the master secret key MK is generated as

$$MK = \langle \beta, g^\alpha \rangle. \tag{3.6}$$

and the public key PK is published as

$$PK = \langle \mathbb{G}_0, g, h = g^\beta, f = g^{\beta^{-1}}, e(g,g)^\alpha \rangle. \tag{3.7}$$

Encryption. $EncABE_{PK}(M, \mathcal{T})$ encrypts a message M under the access tree $\mathcal{T}$. Choose a polynomial q_x for each node x in $\mathcal{T}$ from top to bottom, starting from root node r as follows. For each node x, set the degree d_x of the polynomial q_x as $d_x = t_x - 1$. For root node r, choose $w \in_R \mathbb{Z}_p$ and set $q_R(0) = w$. For every other node x, set $q_x(0) = q_{parent(x)}(index(x))$. For all nodes, choose d_x other points randomly to define q_x completely.

Let Y be the set of leaf nodes in $\mathcal{T}$. The ciphertext is computed as

$$\begin{aligned} CT = \langle \mathcal{T}, \tilde{C} = M \cdot e(g,g)^{\alpha w}, C = h^w, \\ \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(att_y)^{q_y(0)} \rangle. \end{aligned} \tag{3.8}$$

Key Generation. $KeyGenABE_{MK}(S)$ generates a decryption key that is associated with a set of attributes S. Choose $r \in_R \mathbb{Z}_p$, and for every attribute $j \in S$, choose $r_j \in_R \mathbb{Z}_p$. The decryption key becomes

$$\begin{aligned} SK = \langle D = f^{(\alpha + r)}, \\ \forall j \in S : D_j = g^r \cdot H(j)^{r_j}, D'_j = g^{r_j} \rangle. \end{aligned} \tag{3.9}$$

Decryption. $DecryptABE_{SK}(CT)$ is defined recursively. $DecryptNode_{SK}(CT, x)$ takes as input a ciphertext $CT = \langle \mathcal{T}, \tilde{C}, C, \forall y \in Y : C_y, C'_y \rangle$, and a node x from $\mathcal{T}$. If node x is a leaf node, then let $i = att_x$. If $i \in S$, then

$$\begin{aligned} DecryptNode(CT, SK, x) &= \frac{e(D_i, C_x)}{e(D'_i, C'_x)} \\ &= \frac{e(g^r \cdot H(i)^{r_i}, h^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})} \\ &= e(g,g)^{r q_x(0)}. \end{aligned} \tag{3.10}$$

If $i \notin S$, then $DecryptNode_{SK}(CT, x) = \perp$.

If x is not a leaf node, for every $z \in S_x$ compute $\mathrm{DecryptNode}_{SK}(CT, z)$ and store the output as F_z. If there exists an arbitrary set S_x of size t_x of child nodes z, such that $F_z \neq \perp$, then compute

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{\Delta_i^{S'_x}(0)} \\ &= e(g,g)^{r q_x(0)}. \end{aligned} \tag{3.11}$$

Otherwise, if no such set exists, then $\mathrm{DecryptNode}_{SK}(CT, x) = \perp$.

To decrypt CT, start by decrypting root node r,

$$\begin{aligned} A &= \mathrm{DecryptNode}_{SK}(CT, r) \\ &= e(g,g)^{r q_r(0)} \\ &= e(g,g)^{rs} \end{aligned} \tag{3.12}$$

Finally, compute $\dfrac{\tilde{C}}{e(C,D)A^{-1}} = M$ to obtain the original message M.

3.3.5. Trusted Execution Environment

A Trusted Execution Environment (TEE) is a tamper-resistant isolated area of a processor that operates alongside the OS. It guarantees integrity and confidentiality of the executed code and data. This means that any code and data loaded into the TEE cannot be observed or altered by an attacker controlling a user environment or OS. When data leaves the TEE, it is encrypted and integrity protected. ARM's TrustZone [7] and Intel's Software eXtension Guard (SGX) [8] are two hardware technologies that can be used to implement such a TEE.

There is no precise definition of a TEE. In general, TEEs offer isolated execution of Trusted Applications (TAs) and secure storage of data and cryptographic keys. An attempt to define a TEE has been made in [42], which states that a TEE is a tamper-resistant processing environment running on a separation kernel. It should guarantee the authenticity and confidentiality of the executed code, and the runtime states should be integrity and confidentiality protected. In addition, the code, data and runtime states should be stored on persistent memory. Further, the TEE should be able to prove its trustworthiness to third parties using remote attestation. Moreover, the content of the TEE can be updated securely, and it should be resistant against software and physical attacks performed on the main memory of the system.

Additionally, GlobalPlatform [9] has standards specified for TEEs [23]. This standard can be achieved by implementing the following security features:

- Isolation from the Rich Execution Environment: the TAs and their data are separated from the rich environment.
- Isolation from other TAs: the TAs are isolated from each other and also from the TEE itself.
- Application management control: only the authenticated entity can modify the TA and the TEE.
- Identification and binding: enforce authenticity and integrity of the TEE and TAs.
- Trusted storage: to ensure integrity, confidentiality, the data from the TA and TEE are stored securely.
- Trusted access to peripherals: the TEE offers APIs access to trusted peripherals, which are controlled by the TEE.
- State of the art cryptography: such as random number generation, cryptography and monotonic time stamps, which are timestamps that are not affected by the system clock, starts at an arbitrary point and only moves forward.

[7] https://developer.arm.com/ip-products/security-ip/trustzone
[8] https://www.intel.com/content/www/us/en/architecture-and-technology/software-guard-extensions.html
[9] https://globalplatform.org/

Despite this standardization by GlobalPlatform, diverse implementations exists of TEEs that do not (completely) conform to these standards, e.g. Intel's SGX. SGX relies on software attestation [16], which proves that a specific piece of code is executed inside a TEE hosted by the trusted hardware (the used processor), which functions as the root of trust. The proof is often a cryptographic signature that certifies the hash of the TEE's contents. The key used to sign the signature is verified against an endorsement certificate created by the trusted hardware's manufacturer. The endorsement certificate states that the key is only known to the trusted hardware, and only used for the purpose of attestation.

In the TEE-PEP and TEE-PACMAN protocols, presented in Chapter 5, we assume that all the devices that are connected to the SMDs are equipped with TEE supported hardware. The SMDs store the key-factors and pseudonym-factors from all the participants, who are relevant to the patient of the SMD, in the TEE. In addition, the code for performing the PEP operations are stored inside the TEE as well, and will be executed there. Only the output will leave the TEE, which is in encrypted form.

3.4. E-healthcare solutions

3.4.1. Polymorphic Encryption and Pseudonymisation

Polymorphic Encryption and Pseudonymisation (PEP) [51] is a framework that handles the exchange of medical data between multiple parties while providing proper access management and security.

An individual is known by a different identifier, a pseudonym, at different parties. Let ID_A denote the identifier of patient A. At party B, A is known as $\mathrm{ID}_A@B = \mathrm{ID}_A^{k_B}$, with $k_B \in_R \mathbb{Z}_p$. Computing $\mathrm{ID}_A@B$ from ID_A is the task of the Transcrypter (TR).

PEP operations

PEP operates on ElGamal encryptions (see Section 3.1). The PEP operations are defined below.

1. Re-randomisation takes an encryption $\langle b, c\rangle$ and a random value r, and produces a randomized encryption of the same message,

$$\mathtt{RandEG}_r(\langle b,c\rangle) := \langle g^r \cdot b, y^r \cdot c\rangle \mod p, \tag{3.13}$$

2. Re-keying takes an encryption $\langle b, c\rangle$ and a key-factor k, and produces an encryption which can be decrypted using secret key $x = x \cdot k$,

$$\mathtt{KeyEG}_k(\langle b,c\rangle) := \langle b^{k^{-1}}, c\rangle \mod p, \tag{3.14}$$

3. Re-shuffling takes an encryption $\langle b, c\rangle$ and a pseudonym-factor s, and produces an encryption which decrypts to M^s,

$$\mathtt{ShuffleEG}_s(\langle b,c\rangle) := \langle b^s, c^s\rangle \mod p. \tag{3.15}$$

4. Re-keyshuffling combines the re-keying and re-shuffling operations,

$$\mathtt{KeyShuffleEG}_{k,s}(\langle b,c\rangle) := \langle b^{sk^{-1}} c^s\rangle \mod p. \tag{3.16}$$

3.4.2. Distributed Polymorphic Encryption and Pseudonymisation

Distributed-PEP (Dist-PEP) [32, 33] is an extension of PEP (see Section 3.4.1). In Dist-PEP, the role of TR is distributed over a group of n TRs, of which a quorum π of t TRs collaborate in order to perform the re-keying and re-shuffling operations. Such a setup is shown in Figure 3.1. Any group of $t-1$ or less TRs is unable to perform these operations or learn secret keys. This is achieved by making use of the fact that PEP's homomorphic operations consist of exponentiations and by using Shamir Secret Sharing [45]. In addition, the TRs take over the role of the Key Server by generating the master secret key and the user secret keys through secret sharing of random values (see Section 3.2.4).

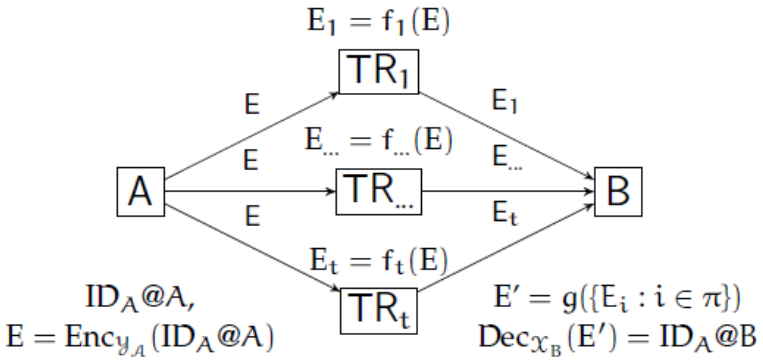

Figure 3.1: Distributed PEP setup [32].

Dist-PEP operations

Consider an ElGamal encryption $\langle b, c\rangle$ and a quorum π of size t. Every $i \in \pi$ receives $\langle b, c\rangle$ and computes a partial result $p_i = \langle b_i, c_i\rangle$, and sends this to the receiving party. Let $P = \{p_i : i \in \pi\}$, the receiving party combines the partial results using

$$\mathtt{CombEG}_\pi(P) := \langle \prod_{i\in\pi} b_i, \prod_{i\in\pi} c_i\rangle. \tag{3.17}$$

A partial result p_i is obtained, given a secret sharing of pseudonym-factor $[s]$ and key-factor $[k]$ as follows.

1. Partial re-randomisation takes an encryption $\langle b, c\rangle$ and a random value r, and produces a partial result of a randomized encryption of the same message,

$$\mathtt{PartRandEG}_r(\langle b, c\rangle) := \langle g^r \cdot b^{t^{-1}}, y^r \cdot c^{t^{-1}}\rangle \mod p, \tag{3.18}$$

2. Partial re-keying takes an encryption $\langle b, c\rangle$ and a secret sharing of a key-factor $[k]$ to pre-compute $[k^{-1}]$, and produces a partial result of an encryption that can be decrypted using secret key $\mathcal{X} = \mathcal{X} \cdot k$,

$$\mathtt{PartKeyEG}_{[k]}(\langle b, c\rangle) := \langle b^{[k^{-1}]_i^\pi}, c^{t^{-1}}\rangle \mod p, \tag{3.19}$$

3. Partial re-shuffling takes an encryption $\langle b, c\rangle$ and a secret sharing of a pseudonym-factor $[s]$, and produces a partial result of an encryption that decrypts to M^s,

$$\mathtt{PartShuffleEG}_{[s]}(\langle b, c\rangle) := \langle b^{[s]_i^\pi}, c^{[s]_i^\pi}\rangle \mod p. \tag{3.20}$$

4. Partial re-keyshuffling combines the partial re-keying and partial re-shuffling operations,

$$\mathtt{PartKeyShuffleEG}_{[k],[s]}(\langle b, c\rangle) := \langle b^{[sk^{-1}]_i^\pi}, c^{[s]_i^\pi}\rangle \mod p. \tag{3.21}$$

3.4.3. Polymorphic Access Management

Polymorphic Access Management (PACMAN) [32] removes the need for an access manager by embedding the access structure into the encryption. This is achieved by using Ciphertext-Policy Attribute Based Encryption (CP-ABE) [8]. Only a party holding the right attributes that satisfy the access structure under which the data is encrypted, is able to decrypt the data. The leafs from the access structures are pseudonymised, ensuring the attributes from getting leaked.

Pseudonymised access trees

Let $\mathcal{T}$ be a tree depicting an access structure. Every node x in the tree has a threshold value t_x and a set of children S_x. Every leaf node has an associated attribute $\mathtt{att}_x \in \mathbb{Z}_p$ and every attribute looks different at every party due to pseudonymisation.

To perform pseudonymisation on the attributes, the tree needs to be encrypted first. Let Y be all the leafs, and $T = \{\langle t_x, S_x\rangle : x \in \mathcal{T}\}$ be all the threshold values and sets of children of all nodes in $\mathcal{T}$. Two operations for encrypting and decrypting trees are defined:

1. $\mathsf{EncTree}_y(\mathcal{T})$ encrypts T as $W = \mathsf{EncEG}_y(T)$, and encrypts every leaf y of $\mathcal{T}$ as $eatt_y = \mathsf{EncEG}_y(att_y)$. The encrypted tree becomes $ET = \langle W, \forall y \in Y : eatt_y \rangle$.

2. $\mathsf{DecTree}_{\mathcal{X}}(\langle W, \forall y \in Y : eatt_y \rangle)$ decrypts W and $eatt_y$ for every $y \in Y$ and reconstructs the access tree.

Given an encrypted tree $ET = \mathsf{EncTree}_y(\mathcal{T} = \langle W, \forall y \in Y : eatt_y \rangle)$, where $Y = g^{\mathcal{X}}$. The following three operations are defined which can be used on encrypted trees.

1. Tree re-randomisation takes an encrypted tree ET and a random value r_W and r_y for every y in Y, and produces a randomized encryption of all its elements,

$$\mathsf{RandTree}(\langle W, \forall y \in Y : eatt_y \rangle) := \langle \mathsf{RandEG}_{r_W}(W), \forall y \in Y : \mathsf{RandEG}_{r_y}(eatt_y) \rangle, \tag{3.22}$$

2. Tree re-keying takes an encrypted tree ET and a key-factor k, and produces an encryption which can be decrypted using secret key $\mathcal{X} = \mathcal{X} \cdot k$,

$$\mathsf{KeyTree}_k(\langle W, \forall y \in Y : eatt_y \rangle) := \langle \mathsf{KeyEG}_k(W), \forall y \text{ in} Y : \mathsf{KeyEG}_k(eatt_y) \rangle, \tag{3.23}$$

3. Tree re-shuffling takes an encrypted tree ET and a pseudonym-factor s, and produces an encryption of which the leaves, or attributes of the tree $eatt_y$ decrypt to $\mathsf{DecEG}_{\mathcal{X}}(att_y)^s$,

$$\mathsf{ShuffleTree}_s(\langle W, \forall y \in Y : eatt_y \rangle) := \langle W, \forall y \in Y : \mathsf{ShuffleEG}(eatt_y) \rangle. \tag{3.24}$$

4. Tree re-keyshuffling combines the re-keying and re-shuffling operations,

$$\mathsf{KeyShuffleTree}_{k,s}(\langle W, \forall y \in Y : eatt_y \rangle) := \langle \mathsf{KeyEG}_k(W), \forall_y \in Y : \mathsf{KeyShuffleEG}_{k,s}(eatt_y) \rangle. \tag{3.25}$$

Construction

Setup. Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be multiplicative groups of prime order p, and g be a generator of $\mathbb{G}_0$ and bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. The master secret key is computed by a trusted authority, choosing a random $\mathcal{X}$, α and β. The master secret key becomes

$$MK = \langle \mathcal{X}, \beta, g^{\alpha} \rangle. \tag{3.26}$$

The corresponding public key is

$$PK = \langle \mathbb{G}_0, g, y = g^{\mathcal{X}}, h = g^{\beta}, f = g^{\beta^{-1}}, e(g,g)^{\alpha} \rangle. \tag{3.27}$$

Key Generation. A secret key SK is generated using a set of attributes S, key-factor k and pseudonym-factor s. Choose $\ell \in_R \mathbb{Z}_p$ and for every attribute $j \in S$, choose $r_j \in_R \mathbb{Z}_p$. SK then becomes

$$\begin{aligned} SK = \mathsf{KeyGenPM}_{MK}(k, s, S) = \langle & \mathcal{X}' = \mathcal{X} \cdot k, \\ & D = f^{(\alpha + \ell)k}, \\ & \forall_j \in S : D_j = g^{\ell} \cdot j^{r_j}, D'_j = g^{r_j} \rangle \end{aligned} \tag{3.28}$$

Encryption. Encrypts a message M under the access structure defined in $\mathcal{T}$ by generating random polynomials as in a CP-ABE encryption process (Sec. 3.3). Cipthertext CT is returned as

$$\begin{aligned} \mathsf{EncPM}_{PK}(M, \mathcal{T}) = \langle & ET = \mathsf{EncTree}_y(\mathcal{T}), \\ & \tilde{C} = Me(g,g)^{\alpha w}, C = h^w, \\ & \forall y \in Y : C_y = g^{q_y(0)}, C'_y = att_y^{q_y(0)} \rangle, \end{aligned} \tag{3.29}$$

where $w \in_R \mathbb{Z}_p$.

Decryption. Decrypt an encryption CT by first decrypting ET to obtain the pseudonymised access tree $\mathcal{T}'$. Let γ be the set of attributes of SK. If γ satisfies $\mathcal{T}'$, SK can be used to decrypt root node r as in a CP-ABE decryption process (Sec. 3.3), $A = DecryptNode_{SK}(CT, r)$. Message M is obtained by computing

$$M = \frac{\tilde{C}}{e(C,D)A^{-1}}. \tag{3.30}$$

PACMAN operations

The PACMAN operations are defined below.

1. PACMAN re-randomisation takes an encryption CT, and a random value r, and produces a randomized encryption of the same message,

$$\begin{aligned} RandPM_r(CT) := \langle \underline{ET} = RandTree(ET), \\ \underline{\tilde{C}} = \tilde{C} \cdot e(g,g)^{\alpha r}, \\ \underline{C} = C \cdot h^r, \\ \forall y \in Y : \underline{C_y} = C_y \cdot g^r, \underline{C'_y} = C'_y \cdot att_y^r \rangle. \end{aligned} \tag{3.31}$$

2. PACMAN re-keying takes an encryption CT, and a key-factor k, and produces an encryption which can be decrypted using secret key $\mathcal{X} = \mathcal{X} \cdot k$,

$$\begin{aligned} KeyPM_k(CT) := \langle \underline{ET} = KeyTree_k(ET), \\ \tilde{C}, \\ \underline{C} = C^{k^{-1}}, \\ \forall y \in Y : C_y, C'_y \rangle. \end{aligned} \tag{3.32}$$

3. PACMAN re-shuffling takes an encryption CT, and a pseudonym-factor s, and produces an encryption in which every attribute att_y of the access tree decrypts to att_y^s,

$$\begin{aligned} ShufflePM_s(CT) := \langle \underline{ET} = ShuffleTree_s(ET), \\ \tilde{C}, \\ C, \\ \forall y \in Y : C_y, C'_y \rangle. \end{aligned} \tag{3.33}$$

4. PACMAN re-keyshuffling combines the re-keying and re-shuffling operations,

$$\begin{aligned} KeyShufflePM_{k,s}(CT) := \langle \underline{ET} = KeyShuffleTree_{k,s}(ET), \\ \tilde{C}, \\ \underline{C} = C^{k^{-1}}, \\ \forall y \in Y : C_y, C'_y \rangle. \end{aligned} \tag{3.34}$$

3.4.4. Distributed Polymorphic Access Management

Distributed Polymorphic Access Management (Dist-PACMAN) [32] is a combination of Dist-PEP and PACMAN. It allows the pseudonymisation of access trees and the PACMAN operations to be performed in a distributed manner.

Distributed Pseudonymised Access Trees

The pseudonymisation of access trees in Section 3.4.3 can be performed in a distributed manner. For every party j, a key-factor k and a pseudonym-factor s are shared between the TRs using Shamir Secret Sharing (see Section 3.2) as $[k_j]$ and $[s_j]$ respectively.

Consider a quorum π of t TRs. Every $i \in \pi$ computes a partial result $ET_i = \langle W_i, \forall y \in Y : eatt_{y,i}$. Let $P = \{ET_i : i \in \pi\}$, the receiving party combines the partial results using

$$CombTree_\pi(P) := \langle CombEG_\pi(\{W_i : i \in \pi\}), \forall y \in Y : CombEG_\pi(\{eatt_{y,i} : i \in \pi\}) \rangle. \tag{3.35}$$

Given an encrypted tree $ET = \langle W, \forall y \in Y : eatt_y \rangle$, a partial result ET_i can be obtained as follows.

1. Partial tree re-randomisation takes an encrypted tree ET and a random value r_W and r_y for every $y \in Y$, and produces a partial result of a randomized encryption of all its elements,

$$\begin{aligned}\mathtt{PartRandTree}(\langle W, \forall y \in Y : \mathtt{eatt}_y \rangle) := \langle &\mathtt{PartRandEG}_{r_W}(W), \\ &\forall y \in Y : \mathtt{PartRandEG}_{r_y}(\mathtt{eatt}_y) \rangle,\end{aligned} \tag{3.36}$$

2. Partial tree re-keying takes an encrypted tree ET and a secret sharing of key-factor k, and produces a partial result of an encryption which can be decrypted using secret key $\mathcal{X} = \mathcal{X} \cdot k$,

$$\begin{aligned}\mathtt{PartKeyTree}_{[k]}(\langle W, \forall y \in Y : \mathtt{eatt}_y \rangle) := \langle &\mathtt{PartKeyEG}_{[k]}(W), \\ &\forall y \ \mathrm{in} Y : \mathtt{PartKeyEG}_{[k]}(\mathtt{eatt}_y) \rangle,\end{aligned} \tag{3.37}$$

3. Partial tree re-shuffling takes an encrypted tree ET and a secret sharing of pseudonym-factor s and let $W = \langle b, c \rangle$, and produces a partial result of an encryption of which the leaves, or attributes of the tree $\mathtt{eatt}_y$ decrypt to $\mathtt{DecEG}_{\mathcal{X}}(\mathtt{att}_y)^s$,

$$\begin{aligned}\mathtt{PartShuffleTree}_{[s]}(\langle W, \forall y \in Y : \mathtt{eatt}_y \rangle) := \langle &\langle b^{-t}, c^{-t} \rangle, \\ &\forall y \in Y : \mathtt{PartShuffleEG}_{[s]}(\mathtt{eatt}_y) \rangle.\end{aligned} \tag{3.38}$$

4. Partial tree re-keyshuffling combines the partial re-keying and partial re-shuffling operations,

$$\begin{aligned}\mathtt{PartKeyShuffleTree}_{[k],[s]}(\langle W, \forall y \in Y : \mathtt{eatt}_y \rangle) := \langle &\mathtt{PartKeyEG}_{[k]}(W), \\ &\forall_y \in Y : \mathtt{PartKeyShuffleEG}_{[k],[s]}(\mathtt{eatt}_y) \rangle.\end{aligned} \tag{3.39}$$

Construction

The construction of Dist-PACMAN takes as its base the construction of ordinary PACMAN (see Section 3.4.3). Key generation is performed by a group of TRs using Shamir Secret Sharing (see Section 3.2). The master key becomes

$$MK = \langle [\mathcal{X}], [\beta], [\beta^{-1}][g^{\alpha}] \rangle. \tag{3.40}$$

The TRs collaborate and publish the public key

$$PK = \langle \mathbb{G}_0, g, y = g^{\mathcal{X}}, h = g^{\beta}, f = g^{\beta^{-1}}, e(g,g)^{\alpha} \rangle. \tag{3.41}$$

For every party i, the TRs share a key-factor $[k_i]$ and a pseudonym-factor $[s_i]$.

Dist-PACMAN operations

Consider a quorum π of t TRs. Every $i \in \pi$ computes a partial result for the re-keying and re-shuffling operations as $p_i = \langle ET_i, \tilde{C}, C_i, \forall_y \in Y : C_y, C'_y \rangle$. Let $P = \{p_i : i \in \pi\}$. The receiving party combines the partial results using

$$\mathtt{CombPM1}_{\pi}(P) := \langle ET = \mathtt{CombTree}_{\pi}(P), \tilde{C}, C = \prod_{j \in \pi} C_j, \forall_y \in Y : C_y, C'_y \rangle. \tag{3.42}$$

Every $i \in \pi$ computes a partial result for the re-randomization operation as $p_i = \langle ET_i, \tilde{C}_j, C_i, \forall_y \in Y : C_{y,i}, C'_{y,i} \rangle$. Let $P = \{p_i : i \in \pi\}$. The receiving party combines the partial results using

$$\begin{aligned}\mathtt{CombPM2}_{\pi}(P) := \langle &ET = \mathtt{CombTree}_{\pi}(P), \tilde{C} = \prod_{j \in \pi} \tilde{C}, C = \prod_{j \in \pi} C_j, \\ &\forall_y \in Y : C_y = \prod_{j \in \pi} C_{y,j}, C'_y = \prod_{j \in \pi} C'_{y,j} \rangle.\end{aligned} \tag{3.43}$$

Let CT be a ciphertext encrypted under access tree $\mathcal{T}$, with Y are the leaves of $\mathcal{T}$. The partial results p_i can be obtained as follows.

1. Partial PACMAN re-randomisation takes an encryption CT, and a random value r, and produces a partial result of a randomized encryption of the same message,

$$\begin{aligned}\mathtt{PartRandPM}_r(CT) := \langle ET_i = \mathtt{PartRandTree}(ET),\\ \tilde{C}_i = \tilde{C}\cdot e(g,g)^{\alpha r},\\ C_i = C\cdot h^r,\\ \forall y \in Y : C_{y,i} = C_y \cdot g^r, C'_{y,i} = C'_y \cdot att_y^r\rangle.\end{aligned} \tag{3.44}$$

2. Partial PACMAN re-keying takes an encryption CT, and a secret sharing of key-factor $[k]$, and produces a partial result of an encryption which can be decrypted using secret key $\mathcal{X} = \mathcal{X}\cdot k$,

$$\begin{aligned}\mathtt{PartKeyPM}_{[k]}(CT) := \langle ET_i = \mathtt{PartKeyTree}_{[k]}(ET),\\ \tilde{C},\\ C_i = C^{[k^{-1}]_i^\pi},\\ \forall y \in Y : C_y, C'_y\rangle.\end{aligned} \tag{3.45}$$

3. Partial PACMAN re-shuffling takes an encryption CT, and a secret sharing of pseudonym-factor $[s]$, and produces a partial result of an encryption in which every attribute att_y of the access tree decrypts to att_y^s,

$$\begin{aligned}\mathtt{PartShufflePM}_{[s]}(CT) := \langle ET_i = \mathtt{PartShuffleTree}_{[s]}(ET),\\ \tilde{C},\\ C_i = C^{t^{-1}},\\ \forall y \in Y : C_y, C'_y\rangle.\end{aligned} \tag{3.46}$$

4. Partial PACMAN re-keyshuffling combines the partial re-keying and partial re-shuffling operations,

$$\begin{aligned}\mathtt{PartKeyShufflePM}_{k,s}(CT) := \langle \underline{ET} = \mathtt{PartKeyShuffleTree}_{k,s}(ET),\\ \tilde{C},\\ C_i = C^{[k^{-1}]_i^\pi},\\ \forall y \in Y : C_y, C'_y\rangle.\end{aligned} \tag{3.47}$$

4

Increasing Quorum Size

In this chapter, we present Increasing Quorum Size (IQS), an extension to Dist-PEP (Sec. 3.4.2), Dist-PACMAN (Sec. 3.4.4), and the protocols TEE-PEP (Sec. 5.1) and TEE-PACMAN (Sec. 5.2). We introduce an approach that uses Lagrange interpolation to increase the size of the quorum of TRs after setup.

Dist-PEP/PACMAN/TEE-PEP/TEE-PACMAN let multiple parties write and read from external storage without them learning each others' identities and without worrying about leaking information. All parties are known by pseudonyms at different parties, TRs take care of transforming encryptions and passes them on from one party to another. A quorum π of t TRs are required to collaborate to perform these tasks, which is achieved by using Shamir Secret Sharing [45]. According to [32], when the system grows in the number of participants, it is to be expected that more TRs need to be added to improve the bandwidth. As the size of the quorum stays constant, the chance of a malicious quorum collaborating increases. Therefore, by increasing the size of the quorum, we are increasing the trust in the system.

To increase the quorum after setup, we use the "threshold changeable modification by the Lagrange method" [37]. This is a share combining technique based on Lagrange interpolation. We assume the TRs are honest-but-curious and will therefore follow the protocol to delete their old shares honestly after new shares have been generated.

First, a description of the threshold changeable modification protocol is provided. Second, we analyze the computational complexity of changing the size of the quorum and present a security analysis.

4.1. Description of IQS

Suppose the setup of a Shamir Secret Sharing with n = number of TRs and t = the threshold (Sec. 3.2). A quorum π of t TRs are required to participate in increasing t to t' with $t' > t$ for an increased quorum π' ($|\pi| < |\pi'|$). The secret S will remain the same as before increasing the quorum. The threshold increasing protocol works as follows:

1. Each TR $i \in \pi$ selects a random polynomial $g(x)$ of degree at most $t'-1$, such that $g_i(0) = f(i)$. The TR then gives a sub-share $g_i(j)$ to all j for $1 \leq j \leq n$, with n = number of TRs.

2. The following public constants are computed:

$$\Delta_i^{\pi}(0) = \prod_{j \in \pi, j \neq i} \frac{0-j}{i-j} \text{ for all } i \in \pi. \tag{4.1}$$

3. Finally, each TR j $(1 \leq j \leq n)$ deletes its old share, and then combines the sub-shares it has received from the t TRs to compute its new share as follows:

$$\varphi_j = \sum_{i \in \pi} (\Delta_i^{\pi} \cdot g_i(j)). \tag{4.2}$$

Using the Lagrange interpolation method, a quorum π' of t' TRs collaborate to recover the secret S:

$$S = \sum_{j \in \pi'} (\Delta_j^{\pi'} \cdot \varphi_j). \tag{4.3}$$

The correctness proof has been added, which is shown in equation (4.4) [37].

$$\begin{aligned}
\sum_{j\in\pi'}(\Delta_j^{\pi'}(0)\cdot\varphi_j) &= \sum_{j\in\pi'}(\Delta_j^{\pi'}(0)\cdot\sum_{i\in\pi}(\Delta_i^{\pi}(0)\cdot g_i(j))) \\
&= \sum_{j\in\pi'}\sum_{i\in\pi}(\Delta_j^{\pi'}(0)(\Delta_i^{\pi}(0)\cdot g_i(j))) \\
&= \sum_{i\in\pi}\sum_{j\in\pi'}(\Delta_j^{\pi'}(0)(\Delta_i^{\pi}(0)\cdot g_i(j))) \\
&= \sum_{i\in\pi}\sum_{j\in\pi'}(\Delta_i^{\pi}(0)(\Delta_j^{\pi'}(0)\cdot g_i(j))) \\
&= \sum_{i\in\pi}(\Delta_i^{\pi}(0)\cdot\sum_{j\in\pi'}(\Delta_j^{\pi'}(0)\cdot g_i(j))) \\
&= \sum_{i\in\pi}(\Delta_i^{\pi}(0)\cdot g_i(0)) \\
&= \sum_{i\in\pi}(\Delta_i^{\pi}(0)\cdot f(i)) \\
&= f(0) \\
&= S.
\end{aligned} \tag{4.4}$$

4.2. Complexity analysis

We have implemented the protocol using C++ and the GMP library to test its performance. We have tested the protocol with different values for the current threshold t, different values for the new threshold t', and different values for the number of TRs n (See resp. Tables 4.1, 4.2, 4.3). The computation time listed is the time spent by a single TR. In addition, we plotted the results in resp. Figures 4.1, 4.2 and 4.3.

In Table 4.1, we observe that the computation time for Step 1 is similar for different values of the current threshold. This is expected, since the new threshold remains the same and the generation of the polynomial depends on the new threshold. Further, the number of TRs remains the same as well, therefore the TRs in the quorum compute the same number of sub-shares. However, the computation time for Step 2 and 3 does change with the different values of the current threshold. This is expected as the higher the current threshold is, the more sub-shares the TRs have to combine into a new share. The plot in Figure 4.1 shows that the computation time of calculating the new share using the sub-shares is linear to the current threshold.

Table 4.1: Computation time in µs with $n = 50$, $t' = 48$.

t	Step 1 computation time (µs)	Step 2 + 3 computation time (µs)
1	2835	0
4	3029	5
7	2891	10
10	2833	16
13	3386	22
16	2874	28
19	2851	32
22	3299	34
25	3057	34
28	2980	36
31	2893	43
34	3009	47
37	3175	47
40	2857	59
43	2943	65
46	2929	67

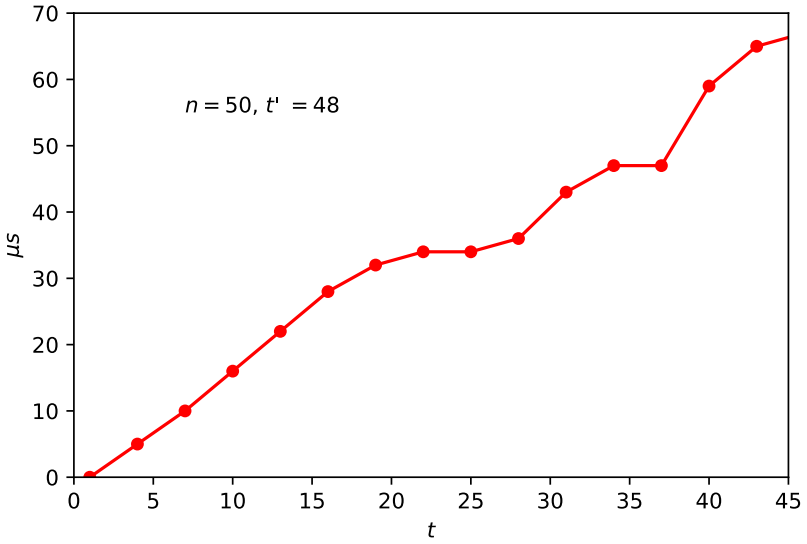

Figure 4.1: Computation time of combining sub shares with a variable value of t.

Table 4.2 shows that the computation time of Step 2 and Step 3 is unaffected by the value differences of the new threshold. Only a fluctuation in the current threshold influences these two steps, as the number of sub-shares a TR has to combine increases with a higher threshold. However, the computation time of Step 1 is influenced by the value of the new threshold, because the generation of the polynomial depends on the new threshold. Hence, the larger the new threshold, the larger the polynomial and the more time it takes to compute the sub-shares. In Figure 4.2, we observe that the computation time of Step 1 is linear to the new threshold.

Table 4.2: Computation time in μs with n = 50, t = 4.

t'	Step 1 computation time (μs)	Step 2 + 3 computation time (μs)
7	124	5
10	166	5
13	221	6
16	304	8
19	317	8
22	511	6
25	829	6
28	1073	7
31	1303	5
34	1491	6
37	1756	5
40	2170	6
43	2348	6
46	2696	6
49	2941	5

In Table 4.3, we notice that the computation time of Step 2 and Step 3 is not influenced by changing the number of TRs. As explained before, this is expected, since Step 2 and Step 3 are only affected by the current threshold. However, the computation time of Step 1 is influenced by the number of TRs. This is because, the larger the number of TRs, the longer it takes for a TR to calculate the sub-shares for all the TRs. It takes longer, because for every TR, the TR in the quorum has to calculate a sub-share. So if the number of TRs increases, the number of sub-shares that the TR in the quorum has to calculate increases as well, which also increases the computation time. The plot in Figure 4.3 illustrates a linear relationship between the computation time of Step 1 and the number of TRs.

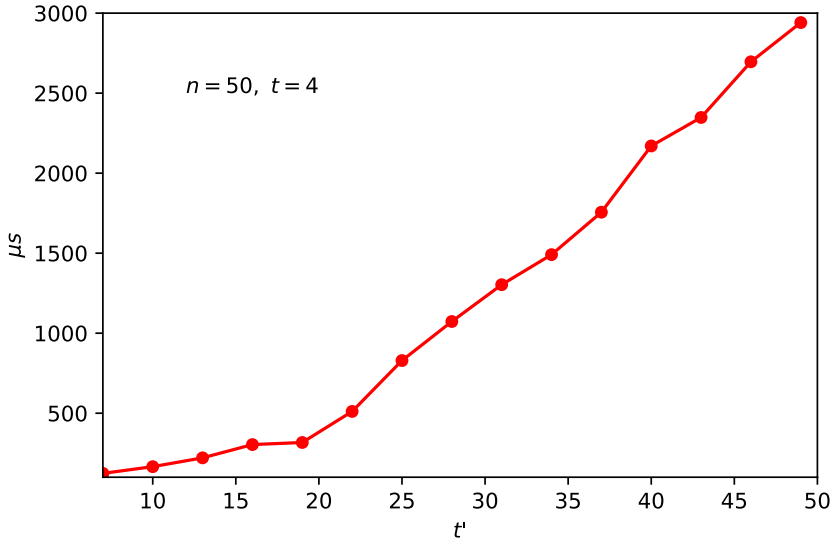

Figure 4.2: Computation time of selecting polynomial and generating sub shares with a variable value of t'.

Table 4.3: Computation time in μs with t = 4, t' = 7.

n	Step 1 computation time (μs)	Step 2 + 3 computation time (μs)
10	39	7
13	33	6
16	45	7
19	51	5
22	48	6
25	69	5
28	76	5
31	75	8
34	85	7
37	90	8
40	98	5
43	102	5
46	106	7
49	121	5
52	127	8

The communication complexity of IQS is measured by the amount of messages sent between the TRs. t TRs construct n subshares and distributes them to n TRs, which leads to $\Theta(tn)$ message exchanges.

This protocol can be applied in Dist-PEP and can be used for increasing the quorum size of the master secret key $[X]$, pseudonym-factors $[s_i]$, and key-factors $[k_i]$ for every participant i. When applied in Dist-PACMAN, IQS can be used for increasing the quorum size of the master key $\langle([X],[\beta],[\beta^{-1}][g^{\alpha}])\rangle$, pseudonym-factors $[s_i]$, and key-factors $[k_i]$ for every participant i.

4.3. Security analysis

In this section, we first give a sketch of the parties in the system with their roles in the IQS protocol. Second, we provide a security analysis of the protocol.

4.3.1. The entities

Patient The IQS protocol does not concern the patient. The master secret key, and the key-factor and pseudonym-factor from the users do not change. The patient does not notice anything.

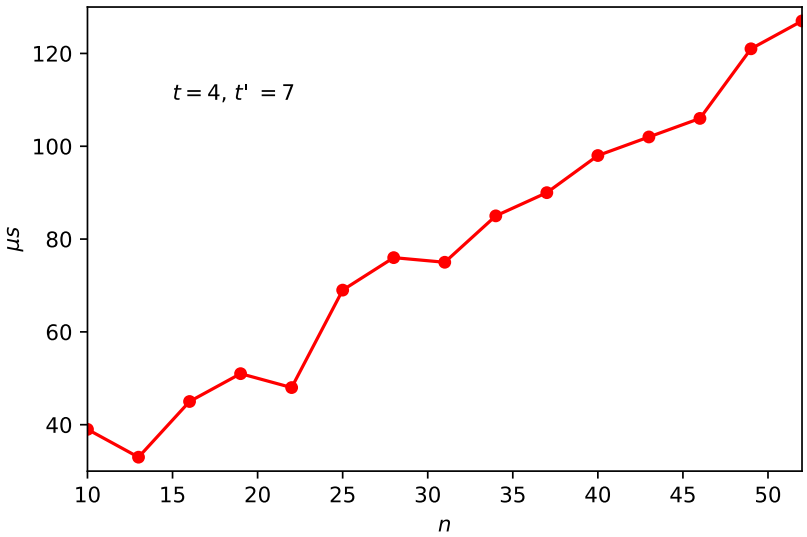

Figure 4.3: Computation time of selecting polynomial and generating sub shares with variable value of n.

Health Care Provider (HCP) The IQS protocol does not concern the HCP. The master secret key, and the key-factor and pseudonym-factor from the users do not change. The HCP does not notice anything.

Transcryptors (TRs) A quorum of t TRs use the IQS protocol to increase the size of the quorum to t', with $t' > t$. The TRs need to use this protocol for every secret sharing of which this quorum is relevant to. The TRs in the quorum use their secret share to create new sub shares. All TRs receive sub shares from the quorum of TRs and combine these sub shares to receive their new share. Because we assume the TRs are honest-but-curious, they delete their old shares by following the protocol. Therefore, a quorum of t or any other number smaller than t' TRs are unable to recover the secret. The TRs need to update the shares from the master secret key and the key-factor and pseudonym-factor from all users.

Self Monitoring Device (SMD) The IQS protocol does not concern the SMD. The master secret key, and the key-factor and pseudonym-factor from the users do not change. The SMD does not notice anything.

Cloud Provider (CP) The IQS protocol does not concern the CP. The master secret key, and the key-factor and pseudonym-factor from the users do not change. The CP does not notice anything.

4.3.2. The protocol

We show that IQS is secure under the passive adversary model, which means that an adversary tries to learn all information possible, but will always follow the protocol specification.

Theorem 1 *The protocol for increasing the quorum of TRs after set-up from* t *to* t'*, where* $t' > t$*, does not leak information about the secret* S *when any set of* $t'-1$ *or less TRs collude.*

It is assumed that the original Shamir Secret Sharing scheme is secure, therefore any set of $t-1$ TRs cannot recover the secret S. A quorum of t TRs re-share their shares using a new random polynomial of degree $t'-1$. This results into any set of $t'-1$ TRs not being able to reconstruct any of these polynomials to reveal the shares from the TRs in the quorum. Since it is assumed that the TRs are honest-but-curious, they will follow the protocol to delete their old shares. This assures that they cannot use any t old shares to compute the secret S. Therefore, the protocol is secure under the passive adversary model.

4.4. Conclusion

By introducing IQS as an extension to Dist-PEP/Dist-PACMAN/TEE-PEP/TEE-PACMAN, we decreased the chance of a group of malicious TRs collaborating, making the system more secure. In addition, we have shown that IQS is secure under the passive adversary model. Further, we analyzed the complexity of calculating

new shares for one secret. All the secret shares from the secret sharings of the private keys, key-factors, and pseudonym-factors from the users need to be re-calculated. We observed that increasing the quorum is more expensive as the quorum increases. The performance highly depends on the total number of TRs, the size of the current quorum, the size of the new quorum, and the number of users.

5

Trusted Execution Environment-based e-healthcare

In the current construction of Distributed Polymorphic Encryption and Pseudonymisation (Dist-PEP) [32, 33] (Sec. 3.4.2) and Distributed Polymorphic Access Management (Dist-PACMAN) [32] (Sec. 3.4.4), the Transcryptors (TRs) have an important role. They perform all the computations: key generation, re-encryption, and replacement of all identifiers with pseudonyms.

In this chapter, we propose using tamper-proof hardware to create Trusted Execution Environments (TEEs) on the devices to which the Smart Monitoring Devices (SMDs) are connected, allowing them to take over some of the computations from the TRs. All key generations are performed by the TRs, while the re-encryptions and replacement of identifiers with pseudonyms are executed by the device to which the SMD of the patient is connected. We call the presented protocols Trusted Execution Environment-based Polymorphic Encryption and Pseudonymisation TEE-PEP (Sec. 5.1) and Trusted Execution Environment-based Polymorphic Access Management TEE-PACMAN (Sec. 5.2).

Wearables usually upload their data through a mobile phone application, or through a desktop application to the Cloud Provider (CP). When mentioniong SMD, we refer to the mobile phone or desktop/laptop that have computation power to the wearable.

Figure 5.1 shows the data flow between the different parties.

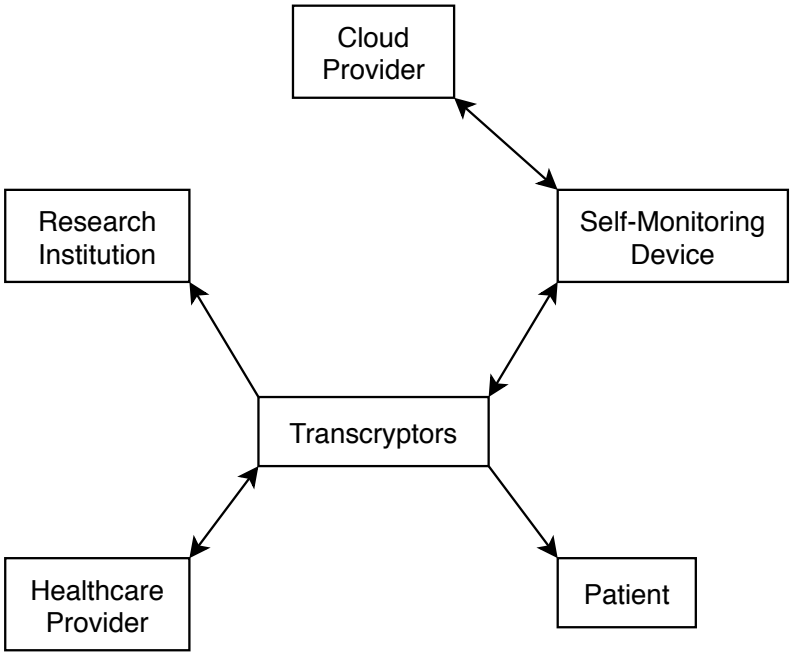

Figure 5.1: Data flow between the different parties.

5.1. TEE-based Polymorphic Encryption and Pseudonymisation

Polymorphic Encryption and Pseudonymisation (PEP) [51] describes a scenario where multiple parties are able to exchange data with the assistance of a trusted third party called the Transcryptor (TR). Dist-PEP and Dist-PACMAN are extensions of this. In our protocol, the PEP operations are performed by the SMDs.

5.1.1. PEP operations

PEP operates on ElGamal encrypted data [21]. Given a private key $\mathcal{X}$ and a public key $\mathcal{Y} = g^{\mathcal{X}}$. Encryption of a message M is performed by

$$\mathrm{EncEG}_{\mathcal{Y}}(M) := \langle g^r, \mathcal{Y}^r \cdot M \rangle \mod p, \text{ where } r \text{ is a random value.} \tag{5.1}$$

To decrypt an encryption $\langle b, c \rangle$,

$$\mathrm{DecEG}_{\mathcal{X}}(\langle b, c \rangle) := b^{\mathcal{X}^{-1}} \cdot c = M \mod p \tag{5.2}$$

is being computed.

The PEP operations are defined below.

1. Re-randomisation takes an encryption $\langle b, c \rangle$ and a random value r, and produces a randomized encryption of the same message,
$$\mathrm{RandEG}_r(\langle b, c \rangle) := \langle g^r \cdot b, \mathcal{Y}^r \cdot c \rangle \mod p, \tag{5.3}$$
2. Re-keying takes an encryption $\langle b, c \rangle$ and a key-factor k, and produces an encryption which can be decrypted using secret key $\mathcal{X} = \mathcal{X} \cdot k$,
$$\mathrm{KeyEG}_k(\langle b, c \rangle) := \langle b^{k^{-1}}, c \rangle \mod p, \tag{5.4}$$
3. Re-shuffling takes an encryption $\langle b, c \rangle$ and a pseudonym-factor s, and produces an encryption which decrypts to M^s,
$$\mathrm{ShuffleEG}_s(\langle b, c \rangle) := \langle b^s, c^s \rangle \mod p, \tag{5.5}$$
4. Re-keyshuffling combines the re-keying and re-shuffling operations,
$$\mathrm{KeyShuffleEG}_{k,s}(\langle b, c \rangle) := \langle b^{sk^{-1}} c^s \rangle \mod p. \tag{5.6}$$

5.1.2. Key generation

The TRs jointly generate the master secret key X as a (t', n) secret sharing of a random value, which is denoted as $[\mathcal{X}]$. The public key $\mathcal{Y}$ is jointly computed by the TRs using $[X]$ and published. The key-factors and pseudonym-factors of the participants are generated in the same manner. In order to generate a (t, n) secret sharing $[S]$ of a random value S, every TR i chooses a random polynomial r_i of degree $t-1$ and gives every TR j a share $r_{i,j}$. Then, every TR j computes their secret share of secret S as $\sum_{i=1}^{n} r_{i,j}$.

For every participant i joining the system, each TR $j \in \pi$ computes $[\mathcal{X} \cdot k_i]_j^{\pi}$ and sends this to i. Participant i then computes its private key as $\mathcal{X}_i = \sum_{j \in \pi} [\mathcal{X} \cdot k_i]_j^{\pi} = \mathcal{X} \cdot k_i$.

5.1.3. Storing and retrieving data

Given an identifier ID_i of a participant i, every participant j learns a pseudonym instead. This pseudonym is denoted by $\mathrm{ID}_i@j = \mathrm{ID}_i^{s_j}$, which is the exponentiation of i's identifier ID_i using j's pseudonym-factor s_j. This process is called pseudonymisation, which is the main operation in PEP. In order to hide the identifiers and pseudonyms from the SMDs, this is performed on an encryption of ID_P.

Setup. The TRs jointly generate the master secret key $\mathcal{X}$ and publishes the public key $\mathcal{Y} = g^{\mathcal{X}}$. For every participant i joining the system, the TRs jointly generate a secret sharing of key-factor $[k_i]$ and pseudonym-factor $[s_i]$, and compute a secret share of the private key $[\mathcal{X} \cdot k_i]$. The shares of the private key are sent to i, who computes its own private key $\mathcal{X}_i$ using the shares.

The TRs send their shares of the pseudonym-factors and key-factors from all participants, who are relevant to the patient, to the patients SMD after performing remote attestation. The SMD computes the pseudonym-factors and key-factors in the TEE using the shares and stores them there, preventing outside code from accessing them.

Storage. Patient P has an SMD, which stores data D in encrypted form ED at a Cloud Provider CP. SMD has the encrypted personal identity $\mathsf{EID_P}$ of P embedded. It re-shuffles and re-keys $\mathsf{EID_P}$ so that CP can decrypt to get $\mathsf{EID_P@CP}$ to store the encrypted data ED there. The re-shuffling and re-keying takes place in its TEE. Figure 5.2 shows this process.

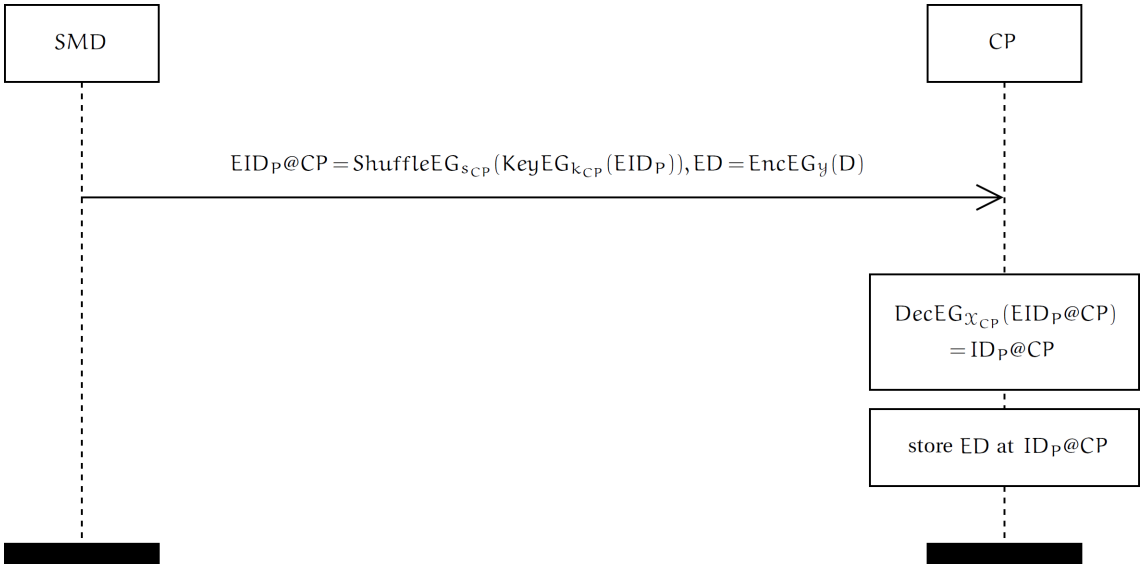

Figure 5.2: Self-Monitoring Device Storing protocol.

When a Doctor A wants to store data D about patient P at CP, A encrypts D to obtain ED and encrypts $\mathsf{ID_P}$ to obtain $\mathsf{EID_P}$, and sends this to the TRs. The TRs relay this to the SMD of patient P. Followed by the SMD re-shuffling and re-keying $\mathsf{EID_P}$ in its TEE to obtain $\mathsf{EID_P@CP}$, and sends both $\mathsf{EID_P@CP}$ and ED to CP. Finally, CP decrypts $\mathsf{EID_P@CP}$ to obtain $\mathsf{EID_P}$, and stores ED there. Figure 5.3 shows this process.

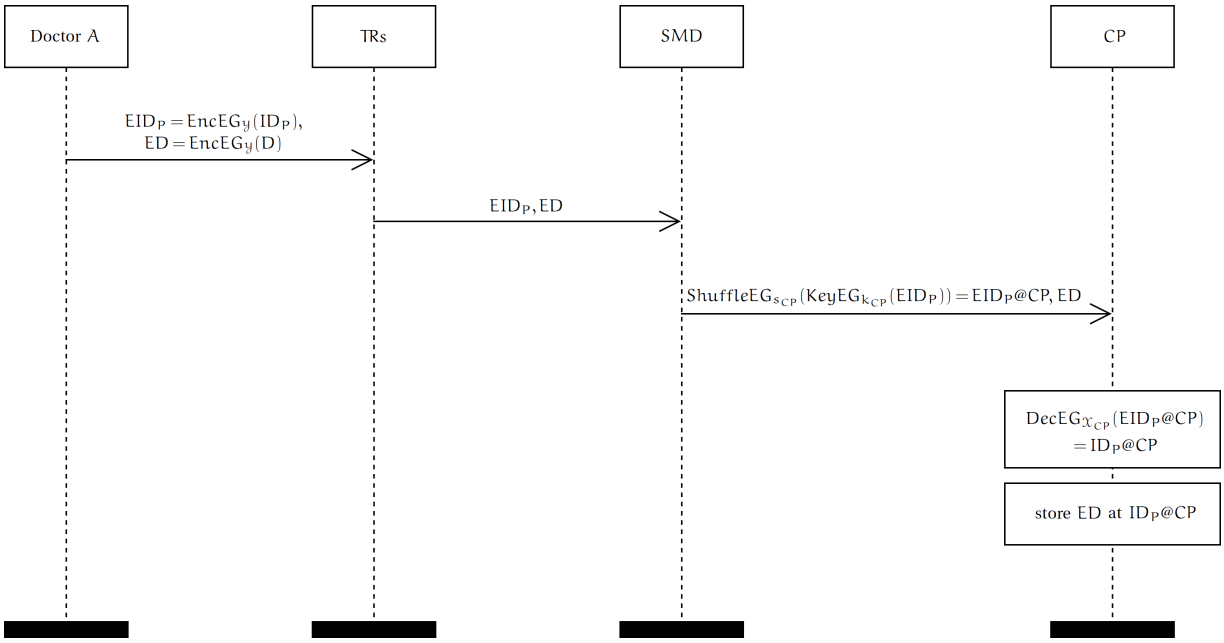

Figure 5.3: Storing protocol.

Retrieval. Patient P has some data D stored at CP in encrypted form ED. When either P or Doctor A wants to retrieve the data, they encrypt $\mathsf{ID_P}$ to obtain $\mathsf{EID_P}$ and sends it to the TRs. TRs relays $\mathsf{EID_P}$ to the SMD from P. Then the SMD re-shuffles and re-keys $\mathsf{EID_P}$ in its TEE to obtain $\mathsf{EID_P@CP}$, and sends this to CP. CP

decrypts EID_P@CP to obtain EID_P and searches for the matching ED. When it finds ED, it re-randomizes ED and sends it to the SMD. SMD re-keys ED in its TEE so that the party asking for the data can decrypt, and sends it to them. Lastly, ED decrypts to D. Figure 5.4 shows this process for Doctor A requesting Patient P's data.

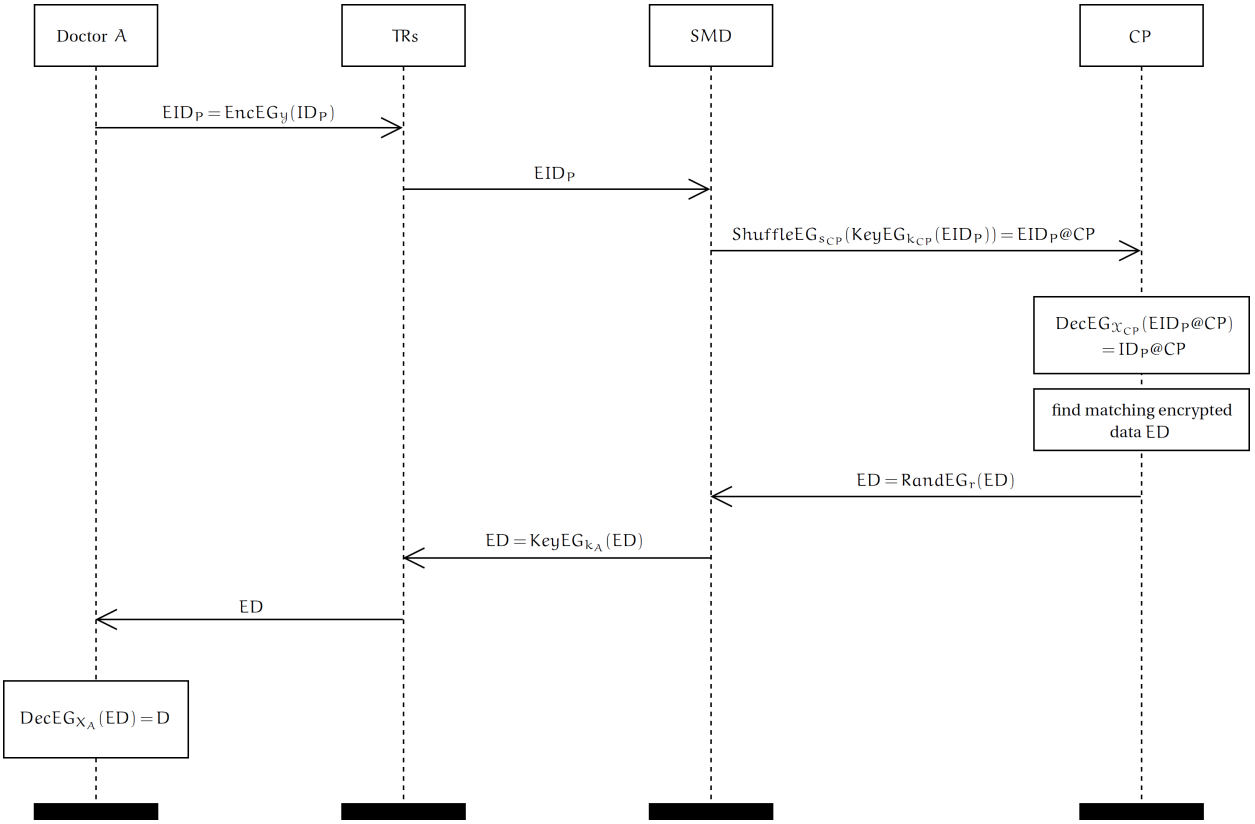

Figure 5.4: Retrieval protocol.

5.2. TEE-based Polymorphic Access Management

Polymorphic Access Managament (PACMAN) [32] describes a scenario where multiple parties are able to exchange data with the help of a group of TRs. Data is encrypted using a modified version of CP-ABE (Sec. 3.3), where the access tree is pseudonymised. This pseudonymisation is performed by the TRs. TEE-PACMAN modifies PACMAN by letting the SMDs perform this pseudonymisation instead.

5.2.1. Pseudonymised access trees

Let $\mathcal{T}$ be a tree depicting an access structure. Every node x in the tree has a threshold value t_x and a set of children S_x. Every leaf node has an associated attribute $att_x \in \mathbb{Z}_p$ and every attribute looks different at every party due to pseudonymisation.

To perform pseudonymisation on the attributes, the tree has to be encrypted first. Let Y be all the leafs, and $T = \{\langle t_x, S_x\rangle : x \in \mathcal{T}\}$ be all the threshold values and sets of children of all nodes in $\mathcal{T}$. Two operations for encrypting and decrypting trees are defined:

1. $\mathsf{EncTree}_y(\mathcal{T})$ encrypts T as $W = \mathsf{EncEG}_y(T)$, and encrypts every leaf y of $\mathcal{T}$ as $eatt_y = \mathsf{EncEG}_y(att_y)$. The encrypted tree becomes $ET = \langle W, \forall y \in Y : eatt_y\rangle$.

2. $\mathsf{DecTree}_{\mathcal{X}}(\langle W, \forall y \in Y : eatt_y\rangle)$ decrypts W and $eatt_y$ for every $y \in Y$ and reconstructs the access tree.

Given an encrypted tree $ET = \mathsf{EncTree}_y(\mathcal{T} = \langle W, \forall y \in Y : eatt_y\rangle)$, where $Y = g^{\mathcal{X}}$. The following three operations are defined which can be used on encrypted trees.

1. Tree re-randomisation takes an encrypted tree ET and a random value r_W and r_y for every y in Y, and produces a randomized encryption of all its elements,

$$\mathtt{RandTree}(\langle W, \forall y \in Y : \mathtt{eatt}_y \rangle) := \langle \mathtt{RandEG}_{r_W}(W), \forall y \in Y : \mathtt{RandEG}_{r_y}(\mathtt{eatt}_y) \rangle, \tag{5.7}$$

2. Tree re-keying takes an encrypted tree ET and a key-factor k, and produces an encryption which can be decrypted using secret key $\mathcal{X} = \mathcal{X} \cdot k$,

$$\mathtt{KeyTree}_k(\langle W, \forall y \in Y : \mathtt{eatt}_y \rangle) := \langle \mathtt{KeyEG}_k(W), \forall y \ in Y : \mathtt{KeyEG}_k(\mathtt{eatt}_y) \rangle, \tag{5.8}$$

3. Tree re-shuffling takes an encrypted tree ET and a pseudonym-factor s, and produces an encryption of which the leaves, or attributes of the tree $\mathtt{eatt}_y$ decrypt to $\mathrm{DecEG}_{\mathcal{X}}(\mathtt{att}_y)^s$,

$$\mathtt{ShuffleTree}_s(\langle W, \forall y \in Y : \mathtt{eatt}_y \rangle) := \langle W, \forall y \in Y : \mathtt{ShuffleEG}(\mathtt{eatt}_y) \rangle, \tag{5.9}$$

4. Tree re-keyshuffling combines the re-keying and re-shuffling operations,

$$\mathtt{KeyShuffleTree}_{k,s}(\langle W, \forall y \in Y : \mathtt{eatt}_y \rangle) := \langle \mathtt{KeyEG}_k(W), \forall_y \in Y : \mathtt{KeyShuffleEG}_{k,s}(\mathtt{eatt}_y) \rangle. \tag{5.10}$$

5.2.2. Key generation

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be multiplicative groups of prime order p, and g be a generator of $\mathbb{G}_0$ and bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. The master secret key is generated by a group of TRs using Secret Sharing of a random value. The master secret key becomes

$$MK = \langle [\mathcal{X}], [\beta], [\beta^{-1}][g^{\alpha}] \rangle. \tag{5.11}$$

From this, the TRs collaborate and publish the public key

$$PK = \langle \mathbb{G}_0, g, y = g^{\mathcal{X}}, h = g^{\beta}, f = g^{\beta^{-1}}, e(g,g)^{\alpha} \rangle. \tag{5.12}$$

For every participant i, the TRs jointly generate a secret sharing of a key-factor $[k_i]$ and pseudonym-factor $[s_i]$.

5.2.3. PACMAN operations

PACMAN operates on PACMAN encryptions, which are similar to CP-ABE encryptions. Let

$$\begin{aligned} SK = \mathtt{KeyGenPM}_{MK}(k, s, S) = \langle \mathcal{X}' = \mathcal{X} \cdot k, \\ D = f^{(\alpha + \ell)k}, \\ \forall_j \in S : D_j = g^{\ell} \cdot j^{r_j}, D'_j = g^{r_j} \rangle \end{aligned} \tag{5.13}$$

be the secret key associated with the set of attributes S, key-factor k and pseudonym-factor s, with $\ell \in_R \mathbb{Z}_p$, and for every attribute $j \in S$, $r_j \in_R \mathbb{Z}_p$.

Encrypting a message M under the access structure defined in $\mathcal{T}$ is done by generating random polynomials as in a CP-ABE encryption process (Sec. 3.3). Ciphertext CT is returned as

$$\begin{aligned} \mathtt{EncPM}_{PK}(M, \mathcal{T}) = \langle ET = \mathtt{EncTree}_y(\mathcal{T}), \\ \tilde{C} = Me(g,g)^{\alpha w}, C = h^w, \\ \forall y \in Y : C_y = g^{q_y(0)}, C'_y = \mathtt{att}_y^{q_y(0)} \rangle, \end{aligned} \tag{5.14}$$

where $w \in_R \mathbb{Z}_p$.

To decrypt an encryption CT, first decrypt ET to obtain the pseudonymised access tree $\mathcal{T}'$. Let γ be the set of attributes of SK. If γ satisfies $\mathcal{T}'$, SK can be used to decrypt root node r as in a CP-ABE decryption process (Sec. 3.3), $A = \mathtt{DecryptNode}_{SK}(CT, r)$. Message M can then be obtained by computing

$$M = \frac{\tilde{C}}{e(C, D)A^{-1}}. \tag{5.15}$$

The three PACMAN operations are defined below.

1. PACMAN re-randomisation takes an encryption CT, and a random value r, and produces a randomized encryption of the same message,

$$\begin{aligned} RandPM_r(CT) := \langle & \underline{ET} = RandTree(ET), \\ & \underline{\tilde{C}} = \tilde{C} \cdot e(g,g)^{\alpha r}, \\ & \underline{C} = C \cdot h^r, \\ & \forall y \in Y : \underline{C_y} = C_y \cdot g^r, \underline{C'_y} = C'_y \cdot att_y^r \rangle. \end{aligned} \tag{5.16}$$

2. PACMAN re-keying takes an encryption CT, and a key-factor k, and produces an encryption which can be decrypted using secret key $\mathcal{X} = \mathcal{X} \cdot k$,

$$\begin{aligned} KeyPM_k(CT) := \langle & \underline{ET} = KeyTree_k(ET), \\ & \tilde{C}, \\ & \underline{C} = C^{k^{-1}}, \\ & \forall y \in Y : C_y, C'_y \rangle. \end{aligned} \tag{5.17}$$

3. PACMAN re-shuffling takes an encryption CT, and a pseudonym-factor s, and produces an encryption in which every attribute att_y of the access tree decrypts to att_y^s,

$$\begin{aligned} ShufflePM_s(CT) := \langle & \underline{ET} = ShuffleTree_s(ET), \\ & \tilde{C}, \\ & C, \\ & \forall y \in Y : C_y, C'_y \rangle. \end{aligned} \tag{5.18}$$

4. PACMAN re-keyshuffling combines the re-keying and re-shuffling operations,

$$\begin{aligned} KeyShufflePM_{k,s}(CT) := \langle & \underline{ET} = KeyShuffleTree_{k,s}(ET), \\ & \tilde{C}, \\ & \underline{C} = C^{k^{-1}}, \\ & \forall y \in Y : C_y, C'_y \rangle. \end{aligned} \tag{5.19}$$

5.2.4. Storing and retrieving data

The access structure is depicted as a tree, with the leaf nodes looking different at every party. Since the internal nodes remain the same, it allows for using the access tree for storage indexing and performing binary search queries.

Setup. The TRs jointly generate the master secret key $MK = \langle [\mathcal{X}], [\beta], [\beta^{-1}][g^\alpha] \rangle$ and publishes the public key $PK = \langle \mathbb{G}_0, g, \mathcal{Y} = g^{\mathcal{X}}, h = g^\beta, f = g^{\beta^{-1}}, e(g,g)^\alpha \rangle$. For every participant i, the TRs jointly generate a secret sharing of a key-factor $[k_i]$ and pseudonym-factor $[s_i]$, and uses these in the generation of the secret key SK.

Storage. Let A be a participant wanting to store data D about patient P at cloud provider CP. A encrypts D under access tree $\mathcal{T}$, $EncPM_{PK}(D, \mathcal{T})$ to obtain CT, and sends this to the TRs. The TRs relay this to the SMD of patient P. Then the SMD re-shuffles and re-keys CT in its TEE, having pseudonymised the encrypted access tree $\mathcal{ET}$ for CP, and sends it to CP. Finally, CP decrypts the access tree and uses it to store CT. Figure 5.5 shows this process for Doctor A storing data D on CP. Note that CP does not have the attributes needed to satisfy the access tree, and can therefore not decrypt the data.

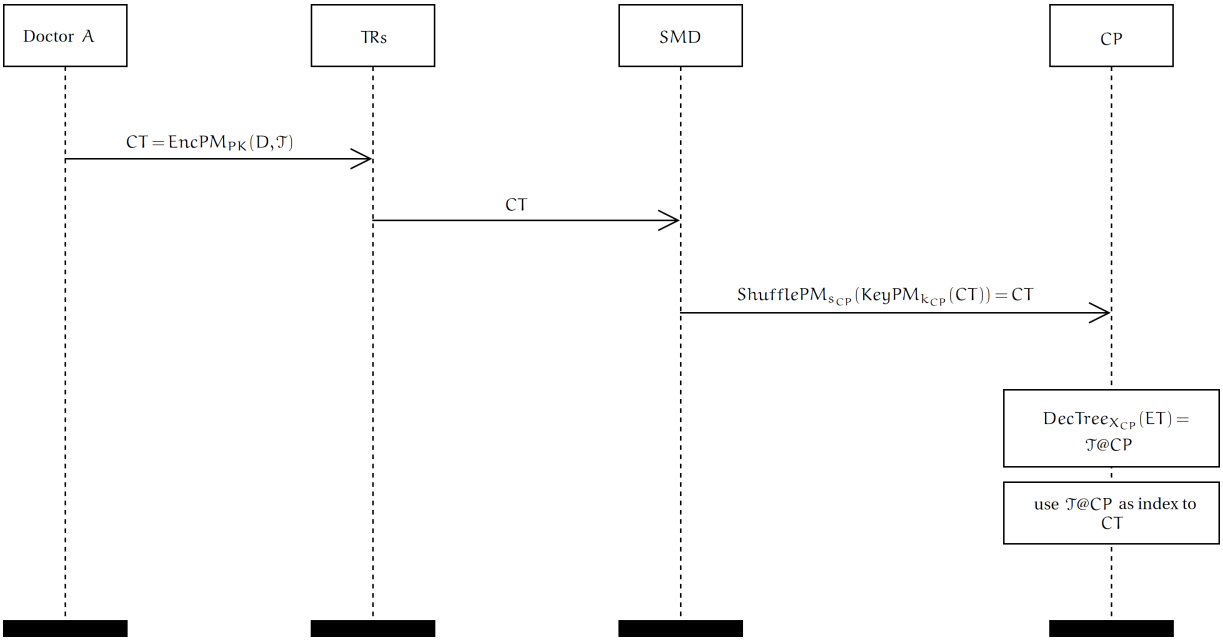

Figure 5.5: Storing protocol.

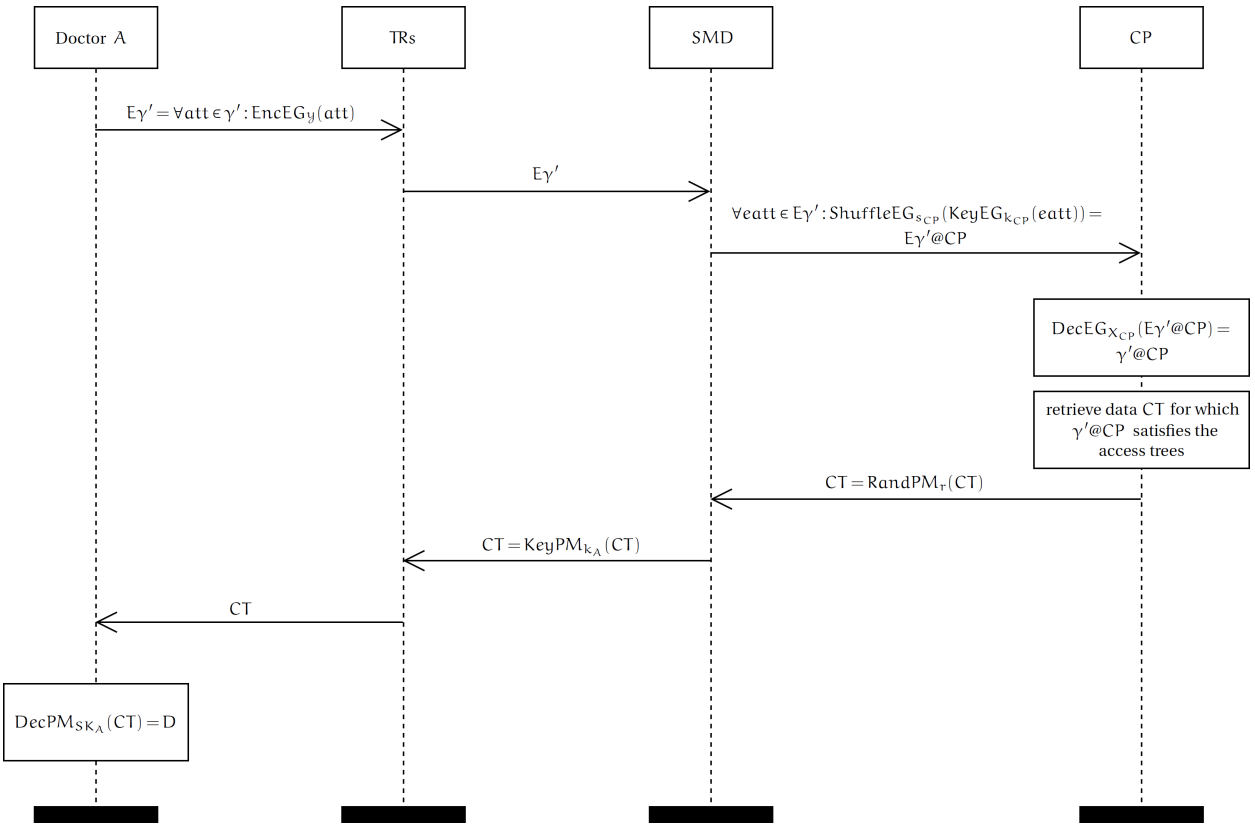

Figure 5.6: Retrieval protocol.

Retrieval. Patient P has data D stored at CP in encrypted form CT. Let B be a doctor with pseudonymised attributes γ wanting to retrieve all data about P from CP which can be satisfied by γ', where $\gamma' \subseteq \gamma$. B encrypts the pseudonymised attributes $att@B \in \gamma'$, and denotes this set of encrypted attributes as $E\gamma'$ and sends this to the TRs. TRs relay this to the SMD of patient P. The SMD re-shuffles and re-keys all $att@B \in \gamma'$

in its TEE, denoted by $E\gamma'@CP$, and sends the result to CP. CP decrypts the attributes in $E\gamma'@CP$ to obtain the set of pseudonymised attributes $\gamma'@CP$ and retrieves all encrypted data CT for which the attributes in $\gamma'@CP$ satisfy the access trees. When it finds CT, it re-randomizes CT and sends it to the SMD. SMD re-keys CT in its TEE so that B can decrypt, and sends it to B. Finally, CT decrypts to D. Figure 5.6 shows this process.

5.3. Analyses

In this section, we analyze the complexity of TEE-PEP and TEE-PACMAN. In addition, we compare the complexity of TEE-PEP with Dist-PEP and TEE-PACMAN with Dist-PACMAN in terms of the PEP and PACMAN operations. Further, we provide a security discussion of our protocols.

5.3.1. Complexity analysis

TEE-PEP

In Table 5.1 a comparison of the computational complexity of the homomorphic operations in terms of multiplications and exponentiations between Dist-PEP and TEE-PEP are shown. The Dist-PEP and TEE-PEP operations are analyzed as if they are performed by a single TR and the combine operation is analyzed for a single participant. The Dist-PEP operations are analyzed for a single TR, because the operations are performed in parallel.

In the table, we observe that TEE-PEP does not have the Combine operation. Since the PEP operations in TEE-PEP are performed by a single SMD, the messages received by the participants do not need to be combined before they can decrypt. Because of this, TEE-PEP has the same computational complexity as PEP.

Table 5.1: Computational complexity Dist-PEP and TEE-PEP compared

Operation	**Dist-PEP [32]**		**TEE-PEP**	
	Mults	Exps	Mults	Exps
Re-Randomize	2	4	2	2
Re-Key	0	2	0	1
Re-Shuffle	0	2	0	2
Re-KeyShuffle	0	2	1	2
Combine	2t	0	—	—

In Dist-PEP a participant sends a polymorphic pseudonym and encrypted data to t TRs. The t TRs perform homomorphic operations on them and return t partial results to a single receiver. Therefore, the message complexity from Dist-PEP is O(t). In TEE-PEP a participant sends a polymorphic pseudonym and encrypted data to t TRs, who forwards them to a single SMD. The SMD performs homomorphic operations on them and returns it to a single receiver. Thus the message complexity is O(t).

TEE-PACMAN

In Table 5.2 a comparison of the computational complexity of the homomorphic operations in terms of multiplications and exponentiations between Dist-PACMAN and TEE-PACMAN are shown. The Dist-PACMAN and TEE-PACMAN operations are analyzed as if they are performed by a single TR and the combine operations are analyzed for a single participant. The Dist-PACMAN operations are analyzed for a single TR, because the operations are performed in parallel.

When comparing the computational complexity of TEE-PACMAN and Dist-PACMAN, we observe that TEE-PEP is more efficient. This is due to TEE-PACMAN not having the CombinePM operations. Since the SMD is performing the PACMAN operations on its own, there is nothing for the participant to combine, and so the computational complexity of the PACMAN operations in TEE-PACMAN is equal to that of PACMAN.

Table 5.2: Computational complexity Dist-PACMAN and TEE-PACMAN compared

Operation	**Dist-PACMAN [32]**		**TEE-PACMAN**	
	Mults	Exps	Mults	Exps
Re-Randomize	2ℓ	$4\ell+2$	2ℓ	$4\ell+2$
Re-Key	0	$2\ell+2$	0	$\ell+1$
Re-Shuffle	0	$2\ell+2$	0	2ℓ
Re-KeyShuffle	ℓ	$2\ell+1$	ℓ	2ℓ
CombinePM1	$2t\ell+\ell$	0	$-$	$-$
CombinePM2	$4t\ell+2\ell$	0		

In Dist-PACMAN a participant sends encrypted data to t TRs. The t TRs perform homomorphic operations on it and return t partial results to a single receiver. Therefore, the message complexity from Dist-PACMAN is O(t). In TEE-PACMAN a participant sends encrypted data to t TRs, who forwards them to a single SMD. The SMD performs homomorphic operations on them and returns it to a single receiver. Thus the message complexity is O(t).

5.3.2. Security analysis

In this section, we give an analysis of the security of the TEE-PEP and TEE-PACMAN protocols. First, we provide a sketch on the actions that each entity is allowed to perform and what information it learns. Second, we provide a security analysis of our protocols.

The entities

Patient A patient can only retrieve data from their own PHR. Depending on which protocol is used, the patient either encrypts their own ID (TEE-PEP), or encrypts their attributes (TEE-PACMAN) and sends it to the TRs. Upon receiving the requested encrypted data back from the TRs, the patient decrypts it using their own private key.

A patient can only retrieve data from their own PHR and has direct contact with only the TRs. Without the private key from other participants in the system, the patient is not able to decrypt data which is not meant for him/her.

Health Care Provider (HCP) A HCP is able to store and retrieve data about a patient. Depending on which protocol is used, the HCP either encrypts the data and the ID from the patient, or encrypts the data under an access tree and sends it to the TRs. The access tree is chosen by the HCP (the encrypter). In TEE-PEP, only someone who has the right private key is able to decrypt, which in this case is the private key from the HCP. In TEE-PACMAN, only someone who holds the right private key with the right attributes is able to decrypt.

Retrieving data from a certain patient from the CP works in the same manner as done by a patient. The only difference is that the HCP encrypts the patients ID, and not their own ID in TEE-PEP, and encrypts their own attributes in TEE-PACMAN.

Transcryptor (TRs) A quorum of TRs in the TEE-PEP and TEE-PACMAN protocols are in charge of key generation and key distribution. They do this in a distributed manner by generating secret sharings of random values and computing their own secret shares. Every participant who joins the system receives pieces of their private key from the TRs, which they need to combine to obtain their real private key. Since this happens in a distributed manner, the security level can be adjusted by increasing the size of the quorum of TRs.

Self Monitoring Device (SMD) Every patient has a SMD. The SMD has a crucial role in the two protocols. The SMD is in charge of performing the PEP and PACMAN operations and is able to store data on the CP. The PEP and PACMAN operations are performed in a TEE, which guarantees correct and secured execution.

Cloud Provider (CP) The cloud provider stores and retrieves encrypted data. Since the data is always encrypted, and the cloud provider does not hold the right keys to the data, it is unable to decrypt the data it has stored.

The protocol

The security of TEE-PEP relies on the security of Dist-PEP and Trusted Execution Environments (TEEs). Dist-PEP's security relies on ElGamal Encryption and Shamir Secret Sharing. Meanwhile, the security of TEE-PACMAN relies on the security of Dist-PACMAN and Trusted Execution Environments (TEEs). Dist-PACMAN's security relies on ElGamal Encryption, Shamir Secret Sharing and Cipthertext-Policy Attribute Based Encryption (CP-ABE).

The security of ElGamal encryption relies on the Discrete Logarithm Problem (DLP), the Computational Diffie-Hellman Problem (DHP) and the Decisional Diffie-Hellman problem (DDH). Let $\mathbb{G}$ be a group of prime order p with generator g.

- DLP: given $g, h \in \mathbb{G}$, it is computationally infeasible to find an integer x such that $g^x = h$.
- DHP: given $g \in \mathbb{G}, a = g^x, b = g^y \in \mathbb{G}$, with $x, y \in_R \mathbb{Z}_p$, it is hard to find c such that $c = g^{xy}$.
- DDH: given $g \in \mathbb{G}, a = g^x, b = g^y, c = g^z$, it is computationally hard to determine whether $z = x \cdot y$.

The security of Shamir Secret Sharing relies on the property that it takes t points to define a polynomial of degree $t-1$. Given $t-1$ points, an infinite number of polynomials of degree $t-1$ exist through these points. This means that given $t-1$ points, an adversary learns nothing about the polynomial, and therefore, learns nothing about the secret.

The security of TEEs rely on the trusted hardware's manufacturer. It relies on certificates from the manufacturer that prove that the CPU is authentic and report measurements of the TEE after launch, which allows verification of the contents of the TEE.

Bilinear maps are required for the construction of CP-ABE. The construction of CP-ABE randomizes users private keys so that they cannot be combined. In order to decrypt, an adversary must recover $e(g,g)^{\alpha s}$. However, the ciphertext is blinded by a value $e(g,g)^{rs}$ and only a user holding the correct key is able to decrypt. Collusions do not work as the blinding value is random to each user's private key.

5.4. Conclusion

In this chapter, we introduced two protocols that extend the Dist-PEP and Dist-PACMAN protocols. By using TEEs in the devices that are connected to the SMDs from the patients, these devices are now performing the PEP operations instead of the TRs. Assuming that the TEEs are manufactured and implemented correctly, it guarantees correct and secure execution of the PEP operations. Through a security analysis, we showed that our protocols do not leak more information than that of Dist-PEP and Dist-PACMAN. Hence, the security of Dist-PEP and Dist-PACMAN is maintained.

Furthermore, through complexity analysis, we observed that our protocols are more efficient. Instead of multiple TRs performing the PEP operations for one patient, only one SMD is performing these operations for one patient. Therefore, there is no need for the patient to combine the results from multiple TRs to obtain the data. The patient can immediately decrypt. Any user only has to combine the sub-shares of their private key once to be able to decrypt.

6

Discussion and future work

In this chapter, we conclude this thesis with a discussion on the contributions to a privacy-preserving electronic healthcare system and propose its future work.

6.1. Discussion

The widely adoption of self-monitoring devices (SMDs) has the potential to improve electronic healthcare, such as by sharing the data gathered by these devices with health care providers (HCPs) to improve diagnosis and treatment of illness. The current Dutch electronic healthcare uses an infrastructure to exchange basic medical information of patients between the HCPs the patient is attending and for use in case of emergencies. However, the patients are not able to inspect their medical dossier at one place and only have the ability to inspect their medical dossiers at each HCP individually. In addition, it does not allow the patients to contribute to their own personal health record (PHR).

To allow patients to store and inspect their own PHR at one place, while securely sharing information with their HCPs using SMDs, the Dist-PEP and Dist-PACMAN protocols facilitate this in a privacy-preserving manner. The encrypted PHR of a patient is stored on a cloud provider. The patient is able to add data gathered by its SMD to its PHR and is able to retrieve data from its PHR. The HCPs that the patient is attending, are able to store and retrieve data from the PHR as well. Dist-PEP is achieved by using Shamir Secret Sharing and homomorphic encryption, while Dist-PACMAN is achieved by using Shamir Secret Sharing, homomorphic encryption and CP-ABE.

Storing data on a cloud provider, while sharing sensitive data in such a manner raises privacy concerns. There are multiple works using homomorphic encryption for processing sensitive data in the medical domain. With the introduction of TEEs in processors, new possibilities for secure management of sensitive data become available. To explore the possibilities of using TEEs in the Dist-PEP and Dist-PACMAN protocols, we formulated the following research question:

How can we use integrated TEEs in SMD-connected devices to reduce the computation load of the TRs and make the PEP framework more efficient and scalable, while preserving privacy and security?

Then we defined two subquestions to aid in answering the research question. The subquestions are:

1. *What other works related to medical data privacy and security are there and how do they make use of TEEs, if any?*
2. *What (cryptographic) methods are there to ensure a secure and privacy-preserving electronic personal health record?*

To answer these subquestions, we explored existing works related to storing and processing medical data and works involving the use of TEEs. We have found a lot of research projects that use encryption in securing and processing private data, especially (fully) homomorphic encryption, as this allows computation on the data while in encrypted form. While encryption provides data confidentiality and integrity, it introduces computation overhead.

Instead of performing computations on encrypted data, TEEs are used to perform computations on decrypted data. Encrypted data enters the TEE and gets decrypted there. Then computations are performed on the data, before it gets encrypted again and leaves the TEE. This is computationally more efficient, since the computations are performed on decrypted data.

In addition, there are works in which both encryption schemes and TEEs are used. Such a solution provides extra protection. If an adversary is successful in breaking the encryption scheme, it is blocked by the TEE. Conversely, if an adversary is successful in breaking the TEE, it is blocked by the encryption scheme.

Further, we have provided the prior art of PEP, Dist-PEP, PACMAN and Dist-PACMAN, which our work is built upon.

As an answer to our research question, we presented TEE-PEP and TEE-PACMAN. We have reduced the computation load of the TRs by moving the execution of the PEP/PACMAN operations from the TRs to the SMD-connected devices. Each SMD-connected device performs the PEP operations inside a TEE, guaranteeing secure execution. Since every patient owns an SMD, and the SMD is only performing the PEP/PACMAN operations on the data concerning the patient, our protocols are more scalable. There is less need to add more TRs to the system, because the TRs do not handle the PEP/PACMAN operations anymore. Furthermore, the users do not need to combine any sub-shares of the data every time before they are able to decrypt the requested data, making the protocols more efficient. The users only need to combine the sub-shares of their private key once, to be able to use it for decryption. Moreover, the privacy guarantees of Dist-PEP and Dist-PACMAN remain unharmed.

Additionally, we provided a description, an implementation and a performance analysis of IQS. IQS is a protocol for increasing the size of the quorum of TRs that are able to perform certain computations. In Dist-PEP and Dist-PACMAN, IQS can be used to increase the quorum size for the generation of keys, key-factors, and pseudonym-factors. In addition, it can be used to increase the quorum size for performing the PEP/PACMAN operations. While in TEE-PEP and TEE-PACMAN, IQS can only be used to increase the quorum size for the generation of keys, key-factors, and pseudonym-factors, as the PEP/PACMAN operations are performed by a single entity.

Our IQS protocol assumes that all parties are honest-but-curious, which means that every participant follows the protocols, but tries to learn all information possible. This is a reasonable assumption in the setting of e-healthcare in The Netherlands, since the regions of the LSP can adopt the role of the TRs. Even if some of them were to be malicious, due to the distributed setting and the minimum number of TRs needed to start the protocol, the threat is reduced.

TEE-PEP and TEE-PACMAN are suitable for use in the Dutch e-healthcare architecture. Currently, the LSP manages the data exchanges between the HCPs. The LSP is divided into 44 regions. These regions can adopt the role of the TRs. The CP can be any CP, because the data is pseudonymised and encrypted. The CP is not able to view the data that it stores and is not able to link the data to a person.

With the introduction of TEEs to the PEP framework, we have managed to improve the efficiency and scalability of the system. In addition, we have introduced an extra layer of protection to the medical data that is being stored at the CP. However, before our protocols can be implemented for use in the Dutch e-healthcare, more research is required.

6.2. Future work

In this work, we introduced the use of tamper-proof hardware to create TEEs for use in the Dist-PEP and Dist-PACMAN protocols. We provided a computational complexity analysis of the PEP operations of Dist-PEP compared to TEE-PEP, and of the PACMAN operations of Dist-PACMAN compared to TEE-PACMAN. As we observed, TEE-PEP and TEE-PACMAN are more efficient. However, future work includes the implementation of a proof-of-concept of the Trusted Execution Environment-based e-healthcare solution. There are many implementations of TEEs possible. A study into which implementation would be best and an actual implementation should provide a better comparison of the efficiency of the protocols. In addition, a more complete security analysis of the used TEE should be provided. Making a decision on which hardware to use for the implementation of the TEE, the availability of the hardware on consumer devices should be considered as well.

We assumed that every patient with an SMD is in the possession of a smartphone or laptop that is equipped with TEE supported hardware to which the SMD is connected to. When a HCP requests to store or retrieve information about the patient, the smartphone or laptop from the patient has to be online to process the request. This is not much of a problem if the request is not urgent. However, in case of an emergency, this

might pose a problem as the HCP is not able to retrieve any information if the SMD-connected device is not online. A solution for HCPs to retrieve information during an emergency should be looked into.

The logging of the data exchanges used to be handled by the TRs, which resulted in a distributed log. This led to challenges in forming consensus and managing the log. With the movement of the task of performing the PEP/PACMAN operations to the SMD, each SMD is able to perform the logging as well. Future work into managing the logging is needed. A comparison of the pros and cons of letting the TRs handle the logging and letting the SMD handle the logging should be made. However, no matter who ends up handling the logging, both parties are able to log the data exchanges.

Another future work that remains, is a study into setting the size of the quorums of TRs. The TRs are still crucial in the current system. The master key, key-factor, and pseudonym-factor generations are handled by the TRs. If the size of the quorum is set too small, the security gets compromised. However, if the size of the quorum is set too large, the efficiency decreases.

Bibliography

[1] Wetsvoorstel 31.466 Elektronisch patiëntendossier, 05 2008. https://www.eerstekamer.nl/wetsvoorstel/31466_elektronisch.

[2] Directive (EU) 2016/680 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data by competent authorities for the purposes of the prevention, investigation, detection or prosecution of criminal offences or the execution of criminal penalties, and on the free movement of such data, and repealing Council Framework Decision 2008/977/JHA, 2016. http://data.europa.eu/eli/dir/2016/680/oj.

[3] Wet op de geneeskundige behandelingsovereenkomst, Art. 7:454 B.W., 09 2020. https://wetten.overheid.nl/BWBR0005290/2020-09-01.

[4] Wet aanvullende bepalingen verwerking persoonsgegevens in de zorg, 07 2020. https://wetten.overheid.nl/BWBR0023864/2020-07-01.

[5] I. Azimiand A.M. Rahmaniand P. Liljeberg and H. Tenhunen. Internet of things for remote elderly monitoring: a study from user-centered perspective. *Journal of Ambient Intelligence and Humanized Computing*, 8(2):273–289, 04 2017. ISSN 1868-5145. doi: 10.1007/s12652-016-0387-y. URL https://doi.org/10.1007/s12652-016-0387-y.

[6] J. Bar-Ilan and D. Beaver. Non-Cryptographic Fault-Tolerant Computing in Constant Number of Rounds of Interaction. In P. Rudnicki, editor, *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing, Edmonton, Alberta, Canada, August 14-16, 1989*, PODC '89, page 201–209. Association for Computing Machinery, 06 1989. ISBN 0897913264. doi: 10.1145/72981.72995. URL https://doi.org/10.1145/72981.72995.

[7] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In J. Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10. Association for Computing Machinery, 01 1988. doi: 10.1145/62212.62213. URL https://doi.org/10.1145/62212.62213.

[8] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-Policy Attribute-Based Encryption. In *2007 IEEE Symposium on Security and Privacy (SP '07), 20-23 May 2007, Oakland, California, USA*, pages 321–334. IEEE Computer Society, 06 2007. doi: 10.1109/SP.2007.11. URL https://doi.org/10.1109/SP.2007.11.

[9] R. Bocu and C. Costache. A homomorphic encryption-based system for securely managing personal health metrics data. *IBM Journal of Research and Development*, 62(1):1:1–1:10, 01 2018. ISSN 0018-8646. doi: 10.1147/JRD.2017.2755524. URL https://doi.org/10.1147/JRD.2017.2755524.

[10] D. Bogdanov. How to securely perform computations on secret-shared data. Master's thesis, University of Tartu, 2007.

[11] J. Bos, K. Lauter, and M. Naehrig. Private Predictive Analysis on Encrypted Medical Data. *Journal of Biomedical Informatics*, 50:234–243, 05 2014. ISSN 1532-0464. doi: 10.1016/j.jbi.2014.04.003. URL https://doi.org/10.1016/j.jbi.2014.04.003.

[12] J.W. Bos, K. Lauter, J. Loftus, and M. Naehrig. Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme. In M. Stam, editor, *Proceedings of the 14th IMA International Conference on Cryptography and Coding, IMACC 2013, Oxford, UK, December 17-19, 2013*, volume 8308 of *Lecture Notes in Computer Science*, page 45–64. Springer, 2013. ISBN 9783642452383. doi: 10.1007/978-3-642-45239-0_4. URL https://doi.org/10.1007/978-3-642-45239-0_4.

[13] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) Fully Homomorphic Encryption without Bootstrapping. In S. Goldwasser, editor, *ITCS '12: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, Cambridge, MA, USA, January 8-10, 2012*, page 309–325. Association for Computing Machinery, 01 2012. ISBN 9781450311151. doi: 10.1145/2090236.2090262. URL https://doi.org/10.1145/2090236.2090262.

[14] Business Insider. Latest trends in medical monitoring devices and wearable health technology, 07 2019. URL https://www.businessinsider.com/wearable-technology-healthcare-medical-devices?international=true&r=US&IR=T.

[15] E. Chiauzzi, C. Rodarte, and P. DasMahapatra. Patient-centered activity monitoring in the self-management of chronic health conditions. *BMC Medicine*, 13(77), 04 2015. ISSN 1741-7015. doi: 10.1186/s12916-015-0319-2. URL https://doi.org/10.1186/s12916-015-0319-2.

[16] V. Costan and S. Devadas. Intel SGX Explained. *IACR Cryptology ePrint Archive*, 2016:86, 2016. URL http://eprint.iacr.org/2016/086.

[17] M. da Rocha, D. Valadares, A. Perkusich, K. Gorgonio, R. Pagno, and N. Will. Secure Cloud Storage with Client-Side Encryption Using a Trusted Execution Environment. In D. Ferguson, M. Helfert, and C. Pahl, editors, *Proceedings of the 10th International Conference on Cloud Computing and Services Science, CLOSER 2020, Prague, Czech Republic, May 7-9, 2020*, volume 1, pages 31–43. SCITEPRESS, 03 2020. ISBN 9789897584244. doi: 10.5220/0009130600310043. URL https://doi.org/10.5220/0009130600310043.

[18] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In G. Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer, 1989. ISBN 978-0-387-34805-6. doi: 10.1007/0-387-34805-0_28. URL https://doi.org/10.1007/0-387-34805-0_28.

[19] Digital Commerce 360. An Apple Watch and app can help see what makes hearts tick, 04 2019. URL https://www.digitalcommerce360.com/2019/04/08/an-apple-watch-and-app-can-help-see-what-makes-hearts-tick/.

[20] N. Drucker and S. Gueron. Achieving trustworthy Homomorphic Encryption by combining it with a Trusted Execution Environment. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, 9(1):86–99, 03 2018. doi: 10.22667/JOWUA.2018.03.31.086. URL https://doi.org/10.22667/JOWUA.2018.03.31.086.

[21] T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1984. ISBN 978-3-540-39568-3. doi: 10.1007/3-540-39568-7_2. URL https://doi.org/10.1007/3-540-39568-7_2.

[22] M. Fonseca, K. Karkaletsis, I. Cruz, A. Berler, and I. Oliveira. OpenNCP: A novel framework to foster cross-border e-Health services. In R. Cornet, L. Stoicu-Tivadar, A. Hörbst, C. L. P. Calderón, S. K. Andersen, and M. Hercigonja-Szekeres, editors, *Digital Healthcare Empowering Europeans - Proceedings of MIE2015, Madrid Spain, 27-29 May, 2015*, volume 210 of *Studies in Health Technology and Informatics*, pages 617–21. IOS Press, 05 2015. doi: 10.3233/978-1-61499-512-8-617. URL https://doi.org/10.3233/978-1-61499-512-8-617.

[23] GlobalPlatform. Introduction to Trusted Execution Environments, 05 2018. https://globalplatform.org/wp-content/uploads/2018/05/Introduction-to-Trusted-Execution-Environment-15May2018.pdf.

[24] P. Gremaud, A. Durand, and J. Pasquier. Privacy-Preserving IoT Cloud Data Processing Using SGX. In *Proceedings of the 9th International Conference on the Internet of Things, IoT 2019, Bilbao, Spain, October 22-25, 2019*, pages 1–4. Association for Computing Machinery, 10 2019. ISBN 978-1-4503-7207-7. doi: 10.1145/3365871.3365888. URL https://doi.org/10.1145/3365871.3365888.

[25] A. Gribov, K. Horan, J. Gryak, K. Najarian, V. Shpilrain, R. Soroushmehr, and D. Kahrobaei. Medical Diagnostics Based on Encrypted Medical Data. In A. Compagnoni, W. Casey, Y. Cai, and B. Mishra, editors, *Bio-inspired Information and Communication Technologies - Proceedings of the 11th EAI International Conference, BICT 2019, Pittsburgh, PA, USA, March 13-14, 2019*, volume 289 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 98–111. Springer, 07 2019. ISBN 978-3-030-24202-2. doi: 10.1007/978-3-030-24202-2_8. URL https://doi.org/10.1007/978-3-030-24202-2_8.

[26] Heart.org. What is Atrial Fibrillation (AFib or AF)?, 07 2016. URL https://www.heart.org/en/health-topics/atrial-fibrillation/what-is-atrial-fibrillation-afib-or-af.

[27] O. Kocabas and T. Soyata. *Towards Privacy-Preserving Medical Cloud Computing Using Homomorphic Encryption*, pages 93–125. 01 2020.

[28] O. Kocabaş and T. Soyata. *Medical data analytics in the cloud using homomorphic encryption*, pages 751–768. IGI Global, 01 2015. ISBN 9781466687578. doi: 10.4018/978-1-4666-8756-1.ch038. URL https://doi.org/10.4018/978-1-4666-8756-1.ch038.

[29] K.A. Küçük, A. Paverd, A. Martin, N. Asokan, A. Simpson, and R. Ankele. Exploring the use of Intel SGX for Secure Many-Party Applications. In *Proceedings of the 1st Workshop on System Software for Trusted Execution, SysTEX@Middleware 2016, Trento, Italy, December 12, 2016*, pages 5:1–5:6. Association for Computing Machinery, 12 2016. ISBN 9781450346702. doi: 10.1145/3007788.3007793. URL https://doi.org/10.1145/3007788.3007793.

[30] X. Li, J. Dunn, D. Salins, G. Zhou, W. Zhou, S.M. Schüssler-Fiorenza Rose, D. Perelman, E. Colbert, R. Runge, S. Rego, R. Sonecha, S. Datta, T. McLaughlin, and M.P. Snyder. Digital Health: Tracking Physiomes and Activity Using Wearable Biosensors Reveals Useful Health-Related Information. *PLOS Biology*, 15(1):1–30, 01 2017. doi: 10.1371/journal.pbio.2001402. URL https://doi.org/10.1371/journal.pbio.2001402.

[31] S. Malwade, S.S.Abdul, M. Uddin, A.A. Nursetyo, L Fernandez-Luque, X. Zhu, L. Cilliers, C. Wong, P. Bamidis, and Y. Li. Mobile and wearable technologies in healthcare for the ageing population. *Computer Methods and Programs in Biomedicine*, 161:233 – 237, 07 2018. ISSN 0169-2607. doi: 10.1016/j.cmpb.2018.04.026. URL https://doi.org/10.1016/j.cmpb.2018.04.026.

[32] C. Maulany. Integrating Self-Monitoring Devices Into the Untrusted Cloud for Healthcare. Master's thesis, Delft University of Technology, 08 2017.

[33] C. Maulany, M. Nateghizad, B. Mennink, and Z. Erkin. Privacy-preserving Distributed Access Control for Medical Data. In *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications, ICETE 2018 - Volume 2: SECRYPT, Porto, Portugal, July 26-28, 2018*, pages 488–497, 2018. doi: 10.5220/0006841404880497. URL https://doi.org/10.5220/0006841404880497.

[34] Ministerie van Volksgezondheid, Welzijn en Sport. VWS Factsheet Wet aanvullende bepalingen verwerking persoonsgegevens in de zorg (Wabvpz), 07 2020. https://www.avghelpdeskzorg.nl/documenten/brochures/2020/07/01/vws-factsheet-wet-aanvullende-bepalingen-verwerking-persoonsgegevens-zorg.

[35] T. Mishra, M. Wang, A.A. Metwally, G.K. Bogu, A.W. Brooks, A. Bahmani, A. Alavi, A. Celli, E. Higgs, O. Dagan-Rosenfeld, B. Fay, S. Kirkpatrick, R. Kellogg, M. Gibson, T. Wang, B. Rolnik, A.B. Ganz, X. Li, and M.P. Snyder. Early Detection Of COVID-19 Using A Smartwatch. *medRxiv*, 07 2020. doi: 10.1101/2020.07.06.20147512. URL https://doi.org/10.1101/2020.07.06.20147512.

[36] A. Natarajan, H.-W. Su, and C. Heneghan. Assessment of physiological signs associated with COVID-19 measured using wearable devices. *medRxiv*, 08 2020. doi: 10.1101/2020.08.14.20175265. URL https://doi.org/10.1101/2020.08.14.20175265.

[37] M. Nojoumian and D.R. Stinson. On Dealer-free Dynamic Threshold Schemes. *Advances in Mathematics of Communications (AMC)*, 7(1):39–56, 01 2013. ISSN 1930-5346. doi: 10.3934/amc.2013.7.39. URL https://doi.org/10.3934/amc.2013.7.39.

[38] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 05 1999. ISBN 978-3-540-48910-8. doi: 10.1007/3-540-48910-X_16. URL https://doi.org/10.1007/3-540-48910-X_16.

[39] Patiëntenfederatie Nederland. Persoonlijke gezondheidsomgeving (PGO). https://www.patientenfederatie.nl/over-de-zorg/pgo.

[40] A. Paverd. *Enhancing communication privacy using trustworthy remote entities*. PhD thesis, University of Oxford, 2015.

[41] A. Paverd, A. Martin, and I. Brown. Privacy-enhanced bi-directional communication in the Smart Grid using trusted computing. In *2014 IEEE International Conference on Smart Grid Communications, SmartGridComm 2014, Venice, Italy, November 3-6, 2014*, pages 872–877. IEEE, 11 2014. ISBN 978-1-4799-4934-2. doi: 10.1109/SmartGridComm.2014.7007758. URL https://doi.org/10.1109/SmartGridComm.2014.7007758.

[42] M. Sabt, M. Achemlal, and A. Bouabdallah. Trusted Execution Environment: What It is, and What It is Not. In *2015 IEEE TrustCom/BigDataSE/ISPA, Helsinki, Finland, August 20-22, 2015*, volume 1, pages 57–64. IEEE, 08 2015. ISBN 978-1-4673-7952-6. doi: 10.1109/Trustcom.2015.357. URL https://doi.org/10.1109/Trustcom.2015.357.

[43] C. Segarra, R. Delgado, M. Lemay, P. Aublin, P. Pietzuch, and V. Schiavoni. Using Trusted Execution Environments for Secure Stream Processing of Medical Data - (Case Study Paper). In J. Pereira and L. Ricci, editors, *Distributed Applications and Interoperable Systems - Proceedings of the 19th IFIP WG 6.1 International Conference, DAIS 2019, Held as Part of the 14th International Federated Conference on Distributed Computing Techniques, DisCoTec 2019, Kongens Lyngby, Denmark, June 17-21, 2019*, volume 11534 of *Lecture Notes in Computer Science*, pages 91–107. Springer, 06 2019. ISBN 978-3-030-22496-7. doi: 10.1007/978-3-030-22496-7_6. URL https://doi.org/10.1007/978-3-030-22496-7_6.

[44] S. Seneviratne, Y. Hu, T. Nguyen, G. Lan, S. Khalifa, K. Thilakarathna, M. Hassan, and A. Seneviratne. A Survey of Wearable Devices and Challenges. *IEEE Communications Surveys Tutorials*, 19(4):2573–2620, 07 2017. ISSN 1553-877X. doi: 10.1109/COMST.2017.2731979. URL https://doi.org/10.1109/COMST.2017.2731979.

[45] A. Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, 11 1979. ISSN 0001-0782. doi: 10.1145/359168.359176. URL https://doi.org/10.1145/359168.359176.

[46] M. Staffa, L. Coppolino, L. Sgaglione, E. Gelenbe, I. Komnios, E. Grivas, O. Stan, and L. Castaldo. KONFIDO: An OpenNCP-based Secure eHealth Data Exchange System. In E. Gelenbe, P. Campegiani, T. Czachórski, S.K. Katsikas, I. Komnios, L. Romano, and D. Tzovaras, editors, *Security in Computer and Information Sciences - First International ISCIS Security Workshop 2018, Euro-CYBERSEC 2018, London, UK, February 26-27, 2018, Revised Selected Papers*, volume 821 of *Communications in Computer and Information Science*, pages 11–27. Springer, 07 2018. ISBN 978-3-319-95189-8. doi: 10.1007/978-3-319-95189-8_2. URL https://doi.org/10.1007/978-3-319-95189-8_2.

[47] Statista. Smartwatch devices unit sales in the United States from 2013 to 2018, November 2018. https://www.statista.com/statistics/381696/wearables-unit-sales-forecast-united-states-by-category/.

[48] Statista. Smartwatch shipments worldwide from 2018 to 2023, December 2019. https://www.statista.com/statistics/878144/worldwide-smart-wristwear-shipments-forecast/.

[49] Statista. Smartwatches - Statistics Facts, May 2019. https://www.statista.com/topics/4762/smartwatches/.

[50] The Verge. Apple Watch electrocardiogram and irregular heart rate features are available today, 12 2018. URL https://www.theverge.com/2018/12/6/18128209/apple-watch-electrocardiogram-ecg-irregular-heart-rate-features-available-health-monitor.

[51] E. Verheul, B. Jacobs, C. Meijer, M. Hildebrandt, and J. de Ruiter. Polymorphic Encryption and Pseudonymisation for Personalised Healthcare. Cryptology ePrint Archive, Report 2016/411, 2016. https://eprint.iacr.org/2016/411.

[52] Volgjezorg. Corona opt-in. https://www.volgjezorg.nl/corona-opt.

[53] M. Zaharia, M. Chowdhury, M.J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster Computing with Working Sets. In E. M. Nahum and D. Xu, editors, *Proceedings of the 2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud'10, Boston, MA, USA, June 22, 2010*, page 10. USENIX Association, 06 2010. URL https://www.usenix.org/conference/hotcloud-10/spark-cluster-computing-working-sets.

[54] M. Zaharia, T. Das, H. Li, S. Shenker, and I. Stoica. Discretized Streams: An Efficient and Fault-Tolerant Model for Stream Processing on Large Clusters. In R. Fonseca and D. A. Maltz, editors, *Proceedings of the 4th USENIX Workshop on Hot Topics in Cloud Computing, HotCloud'12, Boston, MA, USA, June 12-13, 2012*, page 10. USENIX Association, 06 2012. URL https://www.usenix.org/conference/hotcloud12/workshop-program/presentation/zaharia.