DELFT UNIVERSITY OF TECHNOLOGY

INTEGRATING SELF-MONITORING DEVICES INTO THE UNTRUSTED CLOUD FOR HEALTHCARE

TU Delft

INTEGRATING SELF-MONITORING DEVICES INTO THE UNTRUSTED CLOUD FOR HEALTHCARE

CHRISTIAN MAULANY

Thesis in order to obtain the degree of Master of Science

Cyber Security Group
Department of Intelligent Systems
Delft University of Technology

August 2017

Christian Maulany: *Integrating Self-Monitoring Devices Into the Untrusted Cloud for Healthcare,* August 2017

SUMMARY

Over the last few years, healthcare providers have moved to digital healthcare management systems. This electronic approach to healthcare data (e-health) promises numerous benefits over traditional healthcare. The patient can gain more control over which data about him is being gathered, and how different healthcare providers are allowed to share this data. Furthermore, this sharing of medical data would reduce the number of complications due to misinformation.

Parallel to the emergence of e-health, we are seeing the rise of a new related product category. By 2020, the market for wearables is expected to more than double compared to 2016. More specifically, the amount of healthcare wearable shipments is expected to grow from 2.5 million to over 50 million over the same period of time. The current healthcare wearables allow a patient to monitor their health and fitness throughout the day, however, sharing this information with healthcare providers is still complicated. We see that current self-monitoring devices store their data in the cloud. However, the cloud providers do not treat the data they gather as medical data, and instead choose to compromise the data's privacy, by using it for advertisement and analytics.

In this thesis, we propose a new form of medical data management, in which data gathered by self-monitoring devices can securely be shared with healthcare providers. We analyze the privacy and security of e-health and self-monitoring devices, and the state of the art of current medical data security. Finally, we propose two cryptographic methods which could serve as the foundation for the future of e-health, granting the patient privacy, security, and full control over his medical data, even when this data is stored at an untrusted cloud provider.

ACKNOWLEDGMENTS

The last 8 months have been quite an experience. Firstly, I have met a lot of great people, from whom I have learned a lot regarding, academic research, cyber security, cryptography, and much more. This thesis deals with actual problems we are facing now, but even more so in the future, in order to prevent e-health from infringing on our privacy. I feel that with this work, I have been able to contribute to a solution to these problems.

As with all projects, there are many people to thank. I would like to start with Zeki, my thesis supervisor. This thesis would not have been possible without your superior guidance, your motivating insults, and your whining about lack of chocolate cake and a six-pack. Thank you for pushing me, to improve both this project and myself. Besides learning a lot from your advice, working with you has also been a lot of fun. I highly appreciate all the effort you have put into guiding this project.

Zeki's arsenal of PhD students, I wish all of you the best of luck finishing your PhD, despite having to deal with Zeki. I enjoyed our technical discussions, but also coffee breaks, apple time, hamburgers, and other activities. For this, I would like to say to you: "tikkimellaar" (Gamze can translate). Chibuike and Sweet Potato, keep learning Dutch, but also work on your English pronunciation. Brojid and Brosan, keep exercising, focus and control. Zhije, keep bringing amazing Chinese food, and ditching us to have lunch with your girlfriend instead.

None of this would have been possible without the support of my family. Special thanks to my parents, Albert and Marja. Thank you for granting me the opportunity to study at this University. Thank you for your support, your patience, and love, allowing me to obtain this degree.

Finally, lets also thank my fellow master thesis students, otherwise I will never hear the end of it. Thank all of you for distracting me, and making the whole experience more fun.

CONTENTS

1 INTRODUCTION 1
- 1.1 Research Objective . . . 2
- 1.2 Contribution . . . 3
- 1.3 Thesis Outline . . . 3

2 STATE OF E-HEALTH IN THE NETHERLANDS 5
- 2.1 The Dutch LSP . . . 5
- 2.2 Legislation . . . 6
- 2.3 Integration of Self-Monitoring Devices . . . 8
  - 2.3.1 Involved Parties . . . 8
  - 2.3.2 Data Sharing . . . 9
- 2.4 Privacy and Security Concerns . . . 9
- 2.5 Conclusion . . . 12

3 EXISTING CRYPTOGRAPHIC E-HEALTH SOLUTIONS 13
- 3.1 Secure Multi-Party Computation . . . 13
  - 3.1.1 Primitives . . . 14
  - 3.1.2 Private Linear Branchin Programs . . . 17
  - 3.1.3 Applications . . . 17
- 3.2 Symmetric Key Encryption . . . 18
  - 3.2.1 Applications . . . 18
- 3.3 Hybrid Public Key Encryption . . . 18
  - 3.3.1 Applications . . . 19
- 3.4 Proxy Re-Encryption . . . 19
  - 3.4.1 Primitives . . . 19
  - 3.4.2 Applications . . . 20
- 3.5 Attribute-Based Encryption . . . 20
  - 3.5.1 Primitives . . . 21
  - 3.5.2 Applications . . . 21
- 3.6 Searching over Encrypted Data . . . 22
  - 3.6.1 Primitives . . . 22
  - 3.6.2 Applications . . . 23
- 3.7 Conclusion . . . 23

4 A DISTRIBUTED E-HEALTH SYSTEM 27
- 4.1 Polymorphic Encryption and Pseudonymisation . . . 27
  - 4.1.1 PEP operations . . . 27
  - 4.1.2 Storing and Retrieving Data . . . 29
- 4.2 Distributed Polymorphic Encryption and Pseudonymisation . . . 30
  - 4.2.1 Dist-PEP Operations . . . 31
  - 4.2.2 Removing the Trusted Key Server . . . 31

4.2.3 Adding a new transcryptor . . . 31
4.3 Analysis . . . 32
4.3.1 Security . . . 32
4.3.2 Complexity . . . 32
4.3.3 Performance . . . 33
4.4 Conclusion and Discussion . . . 34

5 PATIENT CONTROLLED ACCESS MANAGEMENT 35
5.1 Polymorphic Access Management for Personal Healthcare Records . . . 35
5.2 Preliminaries . . . 38
5.2.1 System Models . . . 38
5.2.2 Security Models . . . 40
5.2.3 Cryptographic building blocks . . . 40
5.3 Polymorphic Access Management . . . 46
5.3.1 Pseudonymised Access Trees . . . 46
5.3.2 Our Construction . . . 47
5.3.3 Performing Pseudonymisation . . . 48
5.3.4 Storing and Retrieving Data . . . 49
5.4 Distributed Polymorphic Attribute-Based Encryption . . . 50
5.4.1 Distributed Pseudonymised Access Trees . . . 50
5.4.2 Our Construction . . . 51
5.4.3 Performing Pseudonymisation . . . 51
5.4.4 Computational Complexity . . . 52
5.4.5 Round Complexity . . . 52
5.4.6 Communication Complexity . . . 54
5.4.7 Storage Complexity . . . 54
5.5 Security Analysis . . . 55
5.5.1 Scheme Security . . . 55
5.5.2 Privacy Analysis . . . 56
5.6 Further improvements . . . 57
5.6.1 Enforcing Access Structures . . . 57
5.6.2 Key Revocation and Time-Based Access Control 58
5.7 Performance Analysis . . . 58
5.8 Application for Medical Data Storage . . . 58
5.8.1 Data Categorization . . . 58
5.8.2 Emergency Attributes . . . 59
5.8.3 Sharing with Research Institutions . . . 59
5.9 Conclusion and Discussion . . . 60

6 CONCLUSION AND DISCUSSION 63
6.1 Future Work . . . 64

A IEEE WIFS 2017 SUBMISSION 67
A.1 Abstract . . . 67
A.2 Introduction . . . 67
A.3 Preliminairies . . . 69

A.3.1 Cryptographic building blocks 69
A.3.2 Polymorhic encryption and Pseudonymization 71
A.4 Distributed Polymorphic Encryption and Pseudonymisation 74
A.4.1 Setup 74
A.4.2 (t, n) PEP Operations 74
A.4.3 Generating new key and pseudonym-factors . . 75
A.4.4 Adding a new transcryptor 75
A.4.5 Key generation 76
A.5 Analysis 76
A.5.1 Security 76
A.5.2 Complexity 76
A.5.3 Performance 77
A.6 Conclusion and Discussion 79

BIBLIOGRAPHY 81

LIST OF FIGURES

Figure 1 Data Flow in PEP 28
Figure 2 PartKeyShuffleEG computation time 33
Figure 3 CombEG computation time 33
Figure 4 Data Flow in System Model 40
Figure 5 PEP setup. 44
Figure 6 Distributed PEP setup. 45
Figure 7 PKRS computation time 78
Figure 8 CPR computation time 78

LIST OF TABLES

Table 1 Symbols and their meaning. 28
Table 2 Computational complexity Dist-PEP operations. 32
Table 3 List of Definitions 39
Table 4 Computational Complexity 53
Table 5 Communication Complexity 54
Table 6 Ciphertext size 54
Table 7 Symbols and their meaning. 73
Table 8 Computational complexity (t, n) PEP operations. 77
Table 9 Computation time of 1000 PRKS and CPR operations 77

ACRONYMS

ABE Attribute-Based Encryption

CP Cloud Provider

CP-ABE Ciphertext-Policy Attribute-Based Encryption

Dist-PEP Distributed Polymorphic Encryption and Pseudonymisation

EHR Electronic Healthcare Record

HCP Healthcare Provider

HSM Hardware Security Module

KP-ABE Key-Policy Attribute-Based Encryption

LBP Linear Bracnhing Program

LSP Landelijk Schakelpunt

PACMAN Polymorphic Access Management

PEP Polymorphic Encryption and Pseudonymisation

PHR Personal Healthcare Record

PKE Public Key Encryption

SKE Symmetric Key Encryption

SMC Secure Multi-party Computations

SMDs Self-Monitoring Devices

TR Transcryptor

TVD Trusted Virtual Domain

1 INTRODUCTION

The market for Self-Monitoring Devices (SMDs) is predicted to surge[1][2], with a predicted market value for wearables of over $ 34 billion by 2020, coming from $ 14 billion in 2016[3][4]. This growth is even more extreme for medical wearables, with shipments increasing from 2.5 million, to more than 55 million over the same period of time[5]. A well known category of SMDs is smartwatches, with examples being Pebble, Apple Watch, and Fitbit. Common in these devices are sensors, such as accelerometers, and heartrate monitors. This arsenal of sensors is likely to increase in the future. Besides smartwatches, we also see more specialized devices, such as wearable glucose monitors, blood-oxygen monitors[6], blood pressure meters, ECG meters[7], and weighing scales.

In general these devices are used for the purpose of fitness tracking, but these devices are starting to cross the line towards medical monitoring[8][9]. One indicator of this development is the type of sensors being used, but also the services provided by the manufacturers for utilizing these measurements. Most SMDs connect to a smartphone using blue-tooth[8], allowing them to upload measurements to the cloud. This cloud storage is generally provided by the manufacturers. Cloud storage improves accessibility and easy of sharing with others. Within the context of fitness tracking, this sharing is usually done with friends, as a form of tracking each others' progress. More medical oriented devices allow sharing with a Healthcare Provider (HCP)[10][11]. At the moment there is no standard for this data sharing. The result is that each manufacturer offers a different method for data sharing.

Expectations are that sharing the self-monitoring data with HCPs will play a huge roll in the future of healthcare[12][8]. This new form of electronic healthcare (e-health) has the potential to increase the quality of healthcare, while reducing its costs. Doctor visits can be replaced by remote monitoring, where SMDs record vital signs and forward them to the HCP[13][14]. Medical personal, or even automated systems monitor the data, and alert the patient if anything out of the ordinary is detected, reducing costs. A proper e-health system would give the patient more insight into his data[15], such as knowledge about what data is being stored, and who is using it. Digitally sharing medical should reduce the amount of complications due to misinformation. Furthermore, if the patient consents, digital data can be anonymised automatically and shared with research institutions, greatly increasing the amount of data at their disposal.

However, we see that the current e-health landscape is not prepared for the use of SMDs. Taking the Netherlands as an example, the exchange of medical data is managed through a central party, the Landelijk Schakelpunt (LSP)[16]. Each patient informs the LSP which HCPs are allowed to exchange his medical data. The LSP acts as a middle man, connecting the HCPs when necessary. In order to connect to the LSP, a HCP needs to be approved, in order to verify they meet certain security standards. Afterwards, they are allowed to request data about a patient at the LSP. The LSP will check if the HCP is authorized, and forward the request to any party able to fulfill it. The LSP is trusted to only forward legitimate requests, and log them. At the moment the patient is not able to digitally access the medical data stored about him. This has to be done at each HCP individually, where the method of doing so may differ. In the current architecture, SMDs have no method of connecting to the LSP, preventing the use of their data for healthcare.

Furthermore, the cloud services provided by the manufacturers are not prepared for the storage of medical data. We see that manufacturers through their privacy policy grant themselves the right to use the stored data for secondary purposes. Examples of such uses are analytics for improving their service, targeted advertisements, and even sharing it with third parties[17][18][19][20][21]. Such practices would not be allowed with medical data[22].

From this we learn that both healthcare and SMDs are not prepared to deliver the benefits promised by e-health. Patients have very limited insight into what data about them is being kept, and limited control over who is sharing this data. Even worse, in order to achieve this limited functionality the system heavily relies on the LSP, a single semi-trusted authority. Furthermore, the current system is not prepared for the emerging trend of wearable and SMDs. The cloud services provided by these devices are unable to connect to the LSP and HCPs, preventing patients from contributing to their own medical records, limiting the potential for innovation and improvements for healthcare. Furthermore the lack of regulations means that currently manufacturers are in a position where they are able to exploit the data gathered by their devices for secondary purposes.

1.1 RESEARCH OBJECTIVE

The main goal of this thesis is to realize secure and privacy-preserving integration of self-monitoring technology into healthcare. We formalize this as the following research question,

How can data gathered by self-monitoring devices be securely shared with healthcare providers, without loss of privacy?

In order to answer this question, we pose the following three sub questions.

- What is the current state of Dutch e-health and how does it make use of SMDs?
- What cryptographic methods are there to provide a secure, privacy-preserving PHR?
- How can the patient stay in control of his medical data?

1.2 CONTRIBUTION

In this thesis we attempt to improve the privacy and security of a patient's data in the current state of e-health. We first provide an analysis of the current state of e-health, by looking at which laws and regulations currently apply. How are HCPs able to exchange medical information, and what control does the patient have over this data, and the role of SMDs. We explore what research exists in order to realize a PHR, where privacy and security is maintained. Finally, we extend the methods of Polymorphic Encryption and Pseudonymisation[23] in two ways. First, we remove the need for a single semi-trusted party and a key server, greatly reducing the security implications when malicious parties collaborate. Second, we remove the need for an access manager, by embedding the access structure into the encryption. Furthermore, we ensure that this access structure leaks no meaningful information. The result is a PHR where a patient, HCPs, SMDs, and cloud providers, can securely exchange a patients medical data. The patient has full control over what types of data each of these parties is able to store and retrieve.

1.3 THESIS OUTLINE

The rest of this thesis is structured as follows. We first answer our first sub question, by providing an analysis of the current state of e-health in the Netherlands in chapter 2. In chapter 3 we explore the second sub question by looking into existing work towards the design of e-health systems. After this we extend this literature with two proposals of our own in chapter 4 and 5. Finally, in chapter 6 we provide a discussion, conclusion, and describe future work.

2

STATE OF E-HEALTH IN THE NETHERLANDS

Over the last decade, we have seen a slow but steady rise of e-health in the Netherlands. HCPs have moved from analog to digital file systems, and a basic infrastructure for exchanging medical data between HCPs is there. However, a PHR, where the patients has full control and insight in his medical data, is far from a reality. Furthermore, we see that the current architecture is not prepared for self-monitoring devices. In this chapter we analyze the current state of e-health in the Netherlands, with the goal of answering our first research question,

What is the current state of Dutch e-health and how does it make use of self-monitoring devices?

In this chapter we describe the state of e-health in the Netherlands, cover relevant legislation, and analyze how self-monitoring devices are integrating into the e-health landscape. We limit the analysis to Dutch e-health, as this allows a more in depth analysis, while we believe this provides a representative picture of e-health in both Europa, and globally. Afterwards, we look into the effect of e-health and the rise of wearables, on the privacy and security of the data involved. Finally, we conclude and attempt to answer our first research question.

2.1 THE DUTCH LSP

The Netherlands has long been struggling to realize the *Electronisch Patiënten Dossier*, an electronic personal healthcare record (PHR). At its inception, this PHR was an effort launched by the Dutch government, and would become a national system where medical data could be stored. This way it would become easier for patients to look into their own medical files, while at the same time granting the patient more control over who has access to their data, and simplifying the way in which medical data could be shared. The motivation being the reduction of medical failure due to misinformation.

Preparations for this project started in 2008, however, already in 2011 work on the project halted due to security and privacy concerns. The parliament was not willing to spend the money needed in order to guarantee the system's security. Though parts of the system are still in use, the government would no longer be involved in its development.

What is left is a decentralized system, called the *Landelijk Schakelpunt* (LSP). In the LSP all HCPs keep their medical files locally, in their own digital systems. The LSP provides a centralized point through which

HCPs can exchange medical data, without storing any medical data itself. All HCPs require an electronic passport, allowing identification, and their file system has to meet certain security standards. Between 90 and 98% of HCPs are using a digital file system connected to the LSP[24]. Chipsoft's HIX and Epic are the most common of these file systems[25].

Each patient has to opt-in to the LSP[26], meaning that by default, the LSP plays no role in their healthcare. Once a patient has opted-in and authorize a HCP to exchange his data, the HCP will notify the LSP, whenever they add new information to the patient's medical file. The LSP in turn will notify any other HCP, to whom this information is relevant, after which the HCPs is able to request this information. The LSP has the responsibility of checking whether this information is relevant for the specific HCP. The patient has access to an online portal, from which it can see which HCPs it has authorized, and when they have exchanged data. The data itself however is not visible, nor can the patient add any information himself. If the patient wishes to see which data is being kept in hise file, it has to be requested at each HCP individually. In general these do not have a digital method of providing this information.

The LSP is managed by *Verniging van Zorgaanbieder voor Zorgcommunicatie* (VZVZ). Funding was originally provided by the Dutch government, however, nowadays funding comes from the HCPs paying a fee to join the LSP, which in turn is covered by health insurance companies. Important to note here is that these health insurance companies do not have access to any data collected by the LSP. Though the LSP does not directly store medical data, they do keep BSN (Dutch equivalent to the Social Security Number of the United States) of each patient, and are aware of all the HCPs the patient has authorized. Furthermore it knows which types of data each HCP is collecting about a patient, and whenever, this information is requested, but not the actual content.

The LSP is divided into several regions. Every HCP is connected to his regional LSP, and is only able to exchange information with other HCPs within the region. Currently most regions are still managed by VZVZ, however management is set to the regions, where responsibility will be given to regional organisations. If a HCP is commonly receiving patients from multiple regions, he is able to connect to multiple regions.

2.2 LEGISLATION

Due to the quick emergence of self-monitoring devices, we see that legislation is not always able to keep up. Legislation for consumer data is much different than medical data, and at the moment self-monitoring devices are placed within a gray area in between. Legis-

lation states that any individual, who's occupation involves working with medical data, should keep this data confidential. This means they are not allowed to share any information about their patients to third parties. By contrast, the data gathered by self-monitoring devices is does not employ this confidentiality. Manufacturers freely sell, share and exploit this data, for purposes such as targeted advertisement and analytics. This implies the data in question is not considered medical data.

This falling behind of legislation is also apparent in the United States, where the Health Insurance Portability and Accountability Act applies[27]. Though this act deals with secure handling of medical data, it does not specify anything about data ownership[28], and state laws are inconsistent. The European Union has recently passed big reform regarding the protection of personal data[22]. As of 25 May 2018, these regulations will take effect. The new regulations are aimed at improving consumer's control over their personal data such as their name, address, cultural profile, income and health information. Companies gathering such information have to abide to stricter rules, such as getting clear consent to process this data, better communications as to why this data is being processed, and provide consumers with a method of obtaining data being kept about them. Furthermore, consumers need to have the right to opt out of any marketing using their data, and should be able to delete their personal data.

Legislation in order to realize the dutch EPR has been going on for quite some years. The BSN is the Dutch equivalent to the United States Social Security Number, a unique number for every Dutch citizen. Use of this number is allowed only by government institutions, and some organizations which have received specific permission through law[29]. Furthermore, such organizations are only allowed to use it for purposes, state in the law. From 2008 medical institutions are allowed to use the BSN to link medical data between different HCPs[30], a first towards the EPD. A motion to enforce all HCPs to connect to the LSP got rejected in 2011. In October 2016, a law passed[31], enforcing extra regulations regarding the electronic exchange of medical data. Effective starting Juli 2017[32], the law prescribes that HCPs are now obligated to inform clients about all electronic data exchange. Note that this electronic data exchange through the LSP is still optional. More importantly, the law gives patients the right to digitally view their medical records. This service has to be provided by the HCPs, free of charge.

In the 2016 Dutch Government Budget[33], E-Health is mentioned as one of the focus policies, in order to improve care, improve consumer experience, and reduce costs. This implies that in the coming years, we can expect to see more legislation changes in order to facility a healthy and hopefully secure e-health environment.

2.3 INTEGRATION OF SELF-MONITORING DEVICES

Several manufacturers of self-monitoring devices and software companies provide a cloud service for storing and managing self-monitoring data. Examples of such services are Apple's HealthKit, Google Fit, Philip's Health Suite, Withings' Health Mate, and Fitbit's dashboard. In this section we explain how these platforms function in general, and how they currently integrate into the dutch e-health.

2.3.1 *Involved Parties*

In the current landscape of e-health and self-monitoring devices, we define the following four parties. The *patient or data owner*, the *cloud provider*, the *healthcare provider* (HCP), the *landelijk schakelpunt* (LSP), and *research institutions*. We give a description of each party, and their role.

- The *patient or data owner* is the subject of all data. Data supplied by the HCP and gathered by self-monitoring devices concerns this individual. The patient informs the LSP which HCPs are authorized to exchange his medical data. The patient has the right to view his medical data at all parties.
- The *cloud provider* is usually the manufacturer of the self-monitoring device, or the company providing the operating system running on these devices. Either way, the cloud provider profits from the sales of the self-monitoring devices. Furthermore, the data itself is of great value, and many cloud providers choose to use this data for secondary purposes.
- The *HCP* gathers and stores medical data about the patient, usually locally in their healthcare system. When given permission, HCPs are able to exchange this medical data with the help of the LSP. Currently there is no standardized method through which HCPs and cloud providers are able to exchange medical data.
- The *LSP* is responsible for facilitating data exchange among HCPs. Furthermore, the LSP monitors misuse, such as HCPs requesting data which should not be relevant for the treatment they provide. The LSP logs any data storage, or data exchange. These logs can be viewed by the patient.
- **Research institutions** wish to use medical data for research purposes. A HCP is allowed to use patient data for personal research. Sharing with research institutions is more restricted. This is only allowed if the patient has given explicit permission, or if receiving permission is not possible in a reasonable manner[34].

Self-monitoring devices can form a great source of data for these instutions. We see that some cloud providers share their data with research institutions. This data does not seem subject to the restrictions put on medical data. Research institutions are not connected to the LSP. Because of this they receive the medical data from HCPs and cloud providers directly.

2.3.2 *Data Sharing*

Several cloud services provide an API through which app developers can access the stored measurements. For this the patient has to give each application permission to access their data, after which the application can retrieve it from the cloud storage. Apple has gone a step further and has partnered with Epic Systems[11], one of the dominant players in the healthcare software market. This partnership is currently running a pilot, where doctors are able to view the patient data stored in Apple's cloud. With this it should be clear that Apple is envisioning a future where wearables and self-monitoring devices are being used for healthcare purposes. By contrast, Google Fit's developer Terms of Service[35], explicitly states that it is not allowed to use Google Fit in connection with any product or service that may qualify as a medical device.

Though some cloud providers store the measurements in encrypted form, it is clear that they themselves are able to decrypt this data. This encryption thus only protects the customer's data in case of a data breach. Further more the privacy statement of all companies we've seen, states that they hold the right to use the data for several purposes, such as analytics, promotions, profiling, and sharing with third parties[17][21][18][20][19]. Furthermore, if the user authorizes third parties to retrieve their personal data from the cloud provider, these third parties usually use their own privacy policy. The often ambiguous descriptions of the privacy policy, and involvement of third parties, makes it difficult to keep track of who is accessing your data, and for which purposes it is being used.

2.4 PRIVACY AND SECURITY CONCERNS

In this section we describe several privacy concerns and security risks, present in the current e-health system in the Netherlands. Furthermore, we review the current state of cloud services provided by self-monitoring devices, and their readiness for the processing of medical data. We first perform an analysis based on the CIA security triad[36]. Afterwards, we discuss more detailed aspects of the current e-health setting, where security might be at risk.

- *Confidentiality* in the current system is limited. For most cloud services, the cloud provider has access to the data being stored. Furthermore, this data is being used for secondary purposes, which would not be allowed for medical data.
- *Integrity* of the data. Exchange of data takes place in encrypted form, ensuring data integrity at this point. In general, the patient file systems employed by the HCPs monitor all actions on their data. Whenever a file gets requested, modified or extended, this action will be logged and linked to the individual performing the action. Because of this, any individual tampering with data can be held accountable.

  For self-monitoring devices, data is gathed at the device, after which it gets uploaded to the cloud provider, usually using the patients smartphone. This gives a slightly more complicated situation, where data integrity may be breached. More importantly, the cloud provider usually has the ability to modify any data stored. For some cloud providers, patients are able to modify and/or manually add measurements.
- *Availability* in the system highly depends on the LSP. The LSP is distributed into smaller regions, however if such a regional LSP goes offline no data can be exchanged between parties within that section. HCPs and cloud providers usually employ proper backups of their data, making loss of data rare.

The BSN is currently used by the LSP and HCPs in order to link patients across different parties. The BSN plays a big role in communication between citizens and government institutions. Unsecure handling of the BSN brings privacy risks, such as identity-theft. Because of this use of the BSN is strictly prescribed by law[29]. Current law does not allow cloud providers to use this BSN for linking patients, thus a different method is.

The LSP in the current setting functions as a semi-trusted authority. The LSP knows all registered HCPs and patient's BSN. Through the LSP the patient authorizes HCPs to exchange his medical data. Because of this the LSP knows the HCPs which a patient is attending, and what type of data these are gathering. Furthermore, the LSP is trusted to only facilitate data exchange accordingly. A malicious LSP would be able to hide data from both patients and HCPs. Furthermore, if it chooses to collaborate with any HCP, the two are able to retrieve all private data. In order to reduce this risk, the LSP is split into several regions. However, this only means that a malicious region is able to learn a smaller subset of the data. One could argue that this regionalisation reduces monitoring of the regions itself, increasing the risk of one operating maliciously.

Bluetooth is usually used in order to facilitate communication between self-monitoring devices and a smartphone[8]. For small wear-

able devices, low energy consumption is usually a priority, the security of these low energy protocols has been shown to be limited[37][38].

Sensitive Data is being collected by the vendors. Profile information gathered usually includes full name, E-mail address, date of birth, gender, weight, height. Furthermore depending on the wearables and smart devices used different metric information will be gathered. These types of measurements include hearth rate, blood pressure, weight, fat percentage, amount of steps taken, location. Usually every measurement can be combined by a note or description given by the patient. In the future we can only expect more types of metrics to be aggregated by these systems. Also permissions granted by the patient to any HCP will also be linked to the patient's data. Besides the fact that identifiable information is often included, also proposedly anonymized data is often still susceptible to linkage attacks[39]. Furthermore the data can be used to learn a lot about individuals, their behaviours and daily routines.

Data Visibility. At the moment the vendor has insight into all the patients data gathered by their products. Furthermore they grant themselves the right to use this data to improve their own services, but also the right to use and sell the data to third parties for e.g. advertisement purposes. The LSP is able to see all HCPs a patient is attending and which types data they are collecting and exchanging. Furthermore in case of a data breach the information gathered by vendors will be visible in plaintext to any attacker.

Data Retention. In the EU, by law individuals have *the right to be forgotten*[40], meaning they should be able to delete any personal data stored at another party. The practicality of this law however, is that fully removing data is not always possible. The privacy agreement of Withings for example, states that a request for data deletion will only result in the removal of the link between patient and the measurement. Furthermore, it is unclear whether data backups will also be altered, whenever an individual employs their right to be forgotten.

Third parties. Many vendors provide ways through which third parties can use the collected data. This third party might have a different privacy policy, rendering guarantees regarding privacy given by the vendor meaningless. The data can be used for any purpose the third party sees fit, such as advertisement or reselling.

Monitoring is mainly handled by the LSP. The LSP makes sure HCPs only exchange relevant information. There is no way for the patinet to verify the information given by the LSP. When a patient requests insight into their file at a HCP, the HCP is the one supplying this data, usually in printed form. This means the HCP is able to tamper and modify the data.

2.5 CONCLUSION

With this analysis we hope to answer the following question,

What is the current state of Dutch e-health and how does it make use of self-monitoring devices?

Over the last decade, the Dutch government has attempted to facility a secure e-health environment, through designing new laws and regulations. However, concerns for the privacy and security implications has often held back these developments. Despite this, we see that slowly the right laws are being formed, in order to facilitate the creation of a PHR. The rudimentary architecture for exchanging medical data between HCPs is there in the form of the LSP. However, we see that the LSP is limited in its functionality. It is able to provide patients with control over who is authorized to share their data, however it can not provide the patient itself access to this data. Because of this patients are unable to view the data stored about them. Furthermore, they are unable to contribute to their PHR, using self-monitoring devices. A big concern with e-health is that or privacy, and with the current architecture we see that that the LSP is given a lot of trust in order to maintain this privacy. The LSP knows for each citizen which HCPs he authorizes to exchange data. However, nothing is stopping the LSP from ignoring this authorization. If the LSP collaborates with a single HCP, he is able to retrieve all medical data, without visibility by the patient. In order to reduce the trust put into the LSP, the LSP is split into several regions.

Regarding the cloud services which are being provided for self-monitoring devices. We see these services are not yet prepared for use in e-health, and as part of the PHR. The laws which ensure the privacy of medical data do not apply to these services yet. We see cloud providers labeling their data as non-medical, allowing them to use it for data mining purposes. The fact that data stored at the cloud provider can be shared with third parties, each with their own privacy policy, makes the situation even more complicated. Cloud providers are not allowed to work with the BSN, further complicating their integration into the Dutch e-health. Only when cloud services start treating the data they handle as medical data, will their integration into the PHR be successful.

3

EXISTING CRYPTOGRAPHIC E-HEALTH SOLUTIONS

Arguable one of the main causes for the slow growth of e-health is the huge security and privacy risks which comes with it. The field of cyber security, however, has long been working on techniques for improving data and communication security. In this chapter we discuss the existing literature relevant to providing a secure e-health environment. For this, we use the distinction between a PHR and a EHR as describe in [41]. A PHR is controlled by patients themselves, while a EHR is controlled by HCPs. Though in this thesis we aim to design a PHR, te current research towards EHR systems will also be of relevance. With the chapter we hope to provide an answer to the following research question,

What cryptographic methods are there to provide a secure, privacy-preserving PHR?

Though different techniques to achieving privacy exist, in this thesis we chose to focus on cryptographic techniques. Since the data has to be viewed by a doctor with a real working relationship with the patient, pure data anonymisation techniques are not applicable to our use-case. Furthermore the provable security of the cryptograhpic techniques, ensure that we can give stronger privacy guarantees about our solution. We discuss several cryptographic approaches, such as Secure Multi-party Computations (SMC), Public Key Encryption (PKE), Attribute-Based Encryption (ABE), and searching over encrypted data. For each approach we give a description of their cryptographic building blocks, and discuss work in which this approach is the main method in order to construct an e-health system. Finally, we shall form an overall conclusion regarding the relevance of these cryptographic methods for the construction of a PHR.

3.1 SECURE MULTI-PARTY COMPUTATION

In SMC, instead of sharing data, a protocol is designed in order to jointly compute a function over the data, while keeping the data private. In the medical context, such an application of SMC usually works in the following manner. A patient holds a dataset of medical measurements, while a service provider holds an algorithm in order to determine if the dataset indicates the presence of a certain illness. The patient does not want to share his data, while the service provider

does not want to reveal his algorithm. This is the core of his service, revealing it would allow anyone to provide this service.

3.1.1 *Primitives*

Several primitives lie at the heart of a wide range of SMC protocols. We describe the most common primitives, and explain how they function.

3.1.1.1 *Oblivious Transfer*

Given a set of messages, oblivious transfer allows Alice to send a subset of these messages to Bob, without learning which messages Bob picks[42]. To demonstrate how this primitive works, we first explain 1 out of 2 oblivious transfer. Alice holds two messages m_0 and m_1, of which one has to be send to Bob, without Alice learning which one Bob receives. For this Alice generates a RSA key pair[43], consisting of modulus N, public exponent e and private exponent d. In addition alice generates two random values x_0 and x_1 and sends these, together with N and e to Bob. Bob picks $b \in \{0, 1\}$, computes $v = x_b + k^e \bmod \prod N$, and sends v to Alice. Alice computes $k_0 = (v - x_0)^d \mod N$, $k_1 = (v - x_1)^d \mod N$, $m'_0 = m_0 + k_0$ and $m'_1 = m_1 + k_1$, which she sends to Bob. Bob computes $m_b = m'_b - k$, obtaining either m_0 or m_1.

This method can be extended to 1 out of n oblivious transfer. Here instead of messages m_0 and m_1, we use messages $m_0, m_{...}, m_{n-1}$, and compute x_i, k_i and m'_i for each message i.

3.1.1.2 *Garbled Circuits*

Garbled Circuits, first proposed by Yao in 1986[44], provide a way by which to parties can jointly evaluate a function over their private inputs. The one limitation being that the function to be evaluated needs to be described as a boolean circuit. One party constructs the garbled circuit while the other party performs the evaluation.

Alice wants to generate a garbled circuit, after which Bob can choose his inputs and evaluate it[45]. For each wire W_i of the circuit Alice generates two secrets, $\widetilde{w}_i^0, \widetilde{w}_i^1$, resulting in garbled wire $\widetilde{W_i} = \langle \widetilde{w}_i^0, \widetilde{w}_i^1 \rangle$. For every gate G_k, Alice generates a garbled table $\widetilde{T}_k$ as follows. Let a and b be the gates input and c its garbled output. Alice generates a truth table for G_k. Since we are dealing with binary gates, this is a 3 column and 4 rows table. She replaces the values for each wire w_i^j by its garbled value $\widetilde{w}_i^j$. Next, for every row in the table she symmetrically encrypts the output value, using the two input values. The result is a garbled table $\widetilde{T}_k$ consisting of 4 rows and 1 column. Alice sends all garbled tables which are part of the circuit to Bob, after which Bob needs to obtain garbled values for his inputs. Sending all garbled inputs would mean Bob can fully decrypt all garbled tables,

leaking the structure of the circuit, while sending only one, would reveal Bob's input. Instead for every input wire, Alice sends only one input to Bob, using oblivious transfer (See Sec. 3.1.1.1). Now for every gate, starting with the input gates, Bob can obtain the garbled output by decrypting each row of the gate. The result can be used as the garbled input for the next gate. The final result will be one or more garbled outputs. Depending on the desired setup, Bob can either share the outputs with Alice, or Alice can share the mapping with Bob.

3.1.1.3 *Homomorphic Encryption*

Homomorphic encryption allows us to perform computations under encryption. To demonstrate, let P be the plaintext space. An additive homomorphic encryption scheme allows us to compute addition in the plaintext domain, by performing corresponding operation $\cdot$ on ciphertext, $\forall x, y \in P : Dec([\![x]\!] \cdot [\![y]\!]) = Dec([\![x + y]\!])$. Examples of such additive cryptoschemes are Paillier[46][47] and Goldwasser-Micali[48].

Besides addition, there are also schemes supporting multiplication under encryption. Examples of such schemes are ElGamal[49] and RSA[43].

3.1.1.4 *Shamir Secret Sharing*

Secret Sharing, first proposed by Shamir in 1979[50], allows a secret to be divided into n pieces. The secret can be retrieved using any t pieces. Let S be the secret to be divided. Pick a random polynomial f of degree $t-1$ such that $f(0) = S$. For every piece i, pick a unique point (x_i, y_i) on f. x_i is made public while y_i is the secret piece.

Given t unique points on f, any point on the polynomial f can be computed using Lagrange interpolation. Let π be a set t unique numbers in $\{1, ..., n\}$. In order to obtain point x we calculate

$$f(x) = \sum_{i \in \pi} \ell_i(x) \mod q, \tag{1}$$

$$\text{where } \ell_i(x) = y_i \prod_{j \in \pi, j \neq s} \frac{x - x_j}{x_i - x_j} \mod q. \tag{2}$$

Since $S = f(0)$, in order to retrieve secret value S we compute $f(0)$.

By giving each point (x_i, y_i) to a different party, value S can be shared among a group of parties. Value S can be retrieved by letting every party compute $S_i = \ell_i(0)$, and combinding the results using Eq. 1, to obtain S. In order to simplify notation, from now on we shall denote the setup of such secret sharing as $[S]$. Next we define the following,

$$\Delta_i^{\pi}(x) = \prod_{j \in \pi, j \neq i} \frac{x - j}{i - j}. \tag{3}$$

Given a quorum π of size t of shares, computing the elements which sum up to S is denoted as,

$$[S]_i^\pi = y_i \Delta_i^\pi(0), \tag{4}$$

$$\sum_{i \in \pi} [S]_i^\pi = S. \tag{5}$$

Given two secret sharings $[S]$ and $[T]$, the parties holding the shares of $[S]$ and $[T]$ can perform computations, of which the result will be a new secret sharing[51]. To compute an addition $[R] = [S + T]$, let $[S]$ and $[T]$ be shared among n parties using polynomials f and g. Let degree of f be $t_f - 1$ and degree of g be $t_g - 1$. $f(0) = S$ and $g(0) = T$. Every party i has a unique public value x_i and holds the corresponding secret values $f(x_i)$ and $g(x_i)$. In order to generate a secret sharing over polynomial $a = f + g$, with $a(0) = f(0) + g(0) = S + T$, every party i computes the value of $a(x_i) = f(x_i) + g(x_i)$ as their secret share of $[R]$. The resulting threshold of $[R]$ is $t_a = \max(t_f, t_g)$.

A secret sharing of $[R] = [S \cdot T]$ can be generated, with threshold t_h. For this a quorum of size $t_f + t_g - 2$ is required. Let π be such a quorum. Every $i \in \pi$ computes $h'(x_i) = f(x_i) \cdot g(x_i)$ locally. This results in a secret sharing of h', with a threshold of $t_f + t_g - 2$. From this a secret sharing with any desired threshold $t_h < t_f + t_g$ can be computed in an interactive manner. For this every $i \in \pi$ creates a secret sharing $h_i = h'(x_i) \prod_{j \in \pi, j \neq i} x_j$, giving every other party j share $h_i(j)$. After exchanging shares, every i computes their secret share of $[R]$ as $\sum_{j \in \pi} h_j(i)$.

A secret sharing of a random value can be generated, For this, every party i picks a random polynomial r_i of degree $t - 1$ and gives every party j a share $r_{i,j}$. Every j computes $\sum_{i \in \pi} r_{i,j}$ as their secret share of $[R]$. The result is a secret sharing with threshold t.

The inverse of a secret, $[R] = [S^{-1} \mod q]$ can also be computed[52]. Let $[S]$ be a t out of n secret sharing using polynomial f with $f(0) = S$. In order to compute the inverse mod q, t parties generate Z, a sharing of a secret random value. Secret shares for $W = Z \cdot S$ are computed. The value of W is revealed and a secret sharing for $[W^{-1} \mod q]$ is generated. Using this, a secret sharing of $[S^{-1} \mod q] = [W^{-1}] \cdot [Z]$ can be calculated.

Given a public value k and a secret sharing $[S]$[53][54][55], we can compute k^S, without revealing S. Given a quorum π, every member i in π calculates $S_i = \ell_i(0)$ using Eq. 2, followed by computing partial result $R_i = k^{S_i}$. These partial results are combined using $\prod_{i \in \pi} R_i = k^S$.

3.1.2 *Private Linear Branchin Programs*

A LBP is a binary decision tree, consisting of z nodes, of which d decision nodes[56]. Every decision node P_i consists of a pair, $P_i = \langle a_i, t_i \rangle$. a_i is the linear combination vector, $a_i = \langle a_{i,1}, .., a_{i,n} \rangle$, t_i is the threshold value. The $z - d$ remaining nodes are the leafs of the tree, forming the classification nodes. Each classification node contains a single classification label. Furthermore, the LBP contains two functions, `Left` and `Right`, which on input of node P_i, output the left or right childnode of P_i respectively.

Given input $x = \langle x_1, .., x_n \rangle$, for each node, starting at the root, we check if $a_i \circ x \leqslant t_i$, which is computes as $\sum_{j=1}^{n} a_{i,j} x_j \leqslant t_i$. If this condition is true we move on the the left child node, otherwise we move to the right child node. Once one of the classification nodes is reached, the output of the LBP is the corresponding classification label.

In order to evalute a LBP in a private manner, a garbled LBP is generated on a server S and send to a client C supplying the inputs. S wants to keep the LBP $\mathcal{K}$ hidden, while C wants to keep its inputs hidden. After execution C learns only the classification label corresponding to his inputs, the amount of decision nodes d in $\mathcal{L}$, and the length of its evaluation path. S learns nothing about C's inputs. The main idea is similar to that of garbled circuits (See Sec. 3.1.1.2. The nodes are garbled, such that the evaluator is able to follow only a single evaluation path, defined by the LBP and the input vector.

The basic principle is as follows. First S creates the garbled LBP $\widetilde{\mathcal{L}}$. Each node P_i is encrypted using two keys, δ_i and k_i. This results in a pair $\widetilde{P}_i$. The evaluator C, can only decrypt one element of this pair. Decryption of element j will return either a pointer to the next node, together with a decryption key k_j, or a classification label. $\widetilde{\mathcal{L}}$ is send to C.

In order to obtain the decryption keys for each node, they employ a protocol called oblivious linear select. In this protocol the comparison of $a_i \circ x \leqslant t_i$ is performed. Depending on the comparison, C learns the decryption key δ_i for one of the elements in $\widetilde{P}_i$. S learns nothing about x and C learns nothing about a_i or t_i.

Now C can evalute the circuit. S sends the decryption key for the root node, k_i and δ_i to C, allowing C to decrypt the root. This will return a pointer to the next node i, together with a decryption key k_i. C can combine k_i and δ_i to decrypt the next node i. This process is repeated until a classification label has been reached.

3.1.3 *Applications*

In [57] this method is used for an ECG classification protocol. In this setup, a service provider has an ECG classification algorithm. A pa-

tient wants to use this algorithm to classify his ECG signal, without revealing his inputs, while the service provider does not want to reveal his algorithm. LBPs can be designed not only for the classification of ECG signals, but can be generalized for many biomedical signal classification algorithms. We first explain how these LBPs work, afterwards we show how these LBPs can be executed in a privacy-preserving manner.

Another common method for data classification are neural networks. In [58][59] a protocol is proposed, where a neural network is able to classify a patients data. Both the data and the neural network are owned by different parties and stay private. In [60] this method is demonstrated for the classification of ECG signals.

In [61], a general classification program is proposed using private binary decision trees. Similar to LBPs this allows for the evaluation of an encrypted signal.

3.2 SYMMETRIC KEY ENCRYPTION

In Symmetric Key Encryption (SKE) the same key is used for encryption and decryption. In order to hide data from the cloud provider, data can be encrypted using SKE before storage. Though applications using SKE usually achieve efficient secure storage, in general these methods require additional measures in order to achieve access control.

3.2.1 *Applications*

An example of an e-health system relying on SKE can be found in [62]. Here data is stored using a symmetric key. A trusted authority hands out smart cards to patients and HCPs. With both a patient and HCP card, a symmetric key is generated which is used for data storage. Though the system achieves secure data storage, the chosen method of key management requires a trusted authority and seems unsuitable for the use of SMDs.

Similarly in [63], SKE is used to achieve patient authorization, data confidentiality, and monitoring. The system again relies on the use of smart cards in order to generate symmetric keys.

3.3 HYBRID PUBLIC KEY ENCRYPTION

A PKE schemes relies on asymmetric encryption in order to achieve security. In PKE different keys are used for encryption and decryption, allowing secure communication. Examples of such forms of encryption are ElGamal[49], RSA[43] and Paillier[46]. Since PKE (in general) is less efficient than SKE, these schemes usually apply a hybrid form

of encryption, where they use symmetric encryption when possible, or use the PKE scheme to encrypt a symmetric key.

3.3.1 *Applications*

In [64], a secure EHR is constructed using the digital rights management approach. Similar to how electronic media is protected, access management for a file, is controlled by the data owner. Though the work achieves strong access control by the patient, it relies on a trusted licence enforcement agent.

The work of [65] uses a combination of PKE and SKE to store medical data at a semi-trusted cloud provider. The system supports anonymous data storage, using a special party to anonymise data. Furthermore, access management can be controlled by the patient. However, the access management relies on the cloud provider to act semi-honest, and breaks if the cloud provider chooses to collaborate with another party. Furthermore, the anonymisation process relies on the help of a trusted party. Because of this, they suggest using medical and government organizations to act as the fully trusted party.

A Trusted Virtual Domain (TVD) aims to achieve privacy and security by running a virtual machine on distributed hardware[66]. A e-health system using TVDs and PKE can be seen in [67]. Though it achieves data privacy, the construction requires authentication by the patient using a PIN, every time data needs to be added to his PHR. Furthermore it requires a very specialized cloud architecture in order to achieve its security.

3.4 PROXY RE-ENCRYPTION

Proxy re-encryption, first introduced by Blaze, Bleumer and Strauss in 1998[54], allows encrypted data to be re-encrypted so that it can be decrypted using a different key. This mechanism can be useful in a number of different applications, with one of the most common being that of delegating access. A patient stores data in encrypted form at a cloud provider. The cloud provider can not access the data, but is able to re-encrypt the data for other parties.

3.4.1 *Primitives*

In order to re-encrypt an encryption for A to an encryption for B, a trusted party either needs the private key of A, or a special re-encryption key, usually computed by A. Many homomorphic encryption schemes allow for re-encryption, such as ElGamal[54], Paillier[68], RSA[69], and lattice-based encryption[70].

Distributed version of proxy re-encryption schemes also exist[71]. In this setup the trusted party is replaced by a group of proxies. The

private key of A, or the re-encryption key is shared among these proxies using Shamir Secret Sharing (See Sec. 3.1.1.4), using threshold t. At least t proxies need to collaborate in order to perform the re-encryption.

3.4.2 *Applications*

In [72] proxy re-encryption is used to reduce key management complexity. Whenever a party is allowed access to a file, the file is re-encrypted for the key of that party. The scheme allows computations to be performed over the encrypted data, however gives no demonstration. Furthermore, access management is completely performed by a fully trusted cloud. Furthermore, as a duplicate of every encryption is kept for every party with access rights, storage complexity is a problem.

In [23] a solution for a PHR is proposed, using a PKE scheme with homomorphic properties. In this work, a central authority replaces identifiers with pseudonyms. These pseudonyms look different at each party, removing linkage of the data owner. Data can be stored in encrypted form, using generic encryption keys. Afterwards the data can be re-encrypted for any party who is allowed to access it. The system achieves hiding of identifiers across different parties, and methods for incorporating self-monitorin devices. The biggest drawbacks of this approach is the reliance on a trusted central authority, access manager, and key server.

Several other works also employ proxy re-encryption as the basis for key and access management[73][74][75][76]. Either the encrypting party, or a key management authority has the responsibility of distributing decryption keys only to authorized parties. This highlights one of the main drawbacks of such an approach. As access rules become more complex, so does key management. Approaches to somewhat reduce this complexity include categorizing files, resulting in less keys to manage, or by introducing a per file access control list.

3.5 ATTRIBUTE-BASED ENCRYPTION

One of the most recent developments is that of ABE[77], first introduced in 2005. ABE shows great promise in its application for PHRs[78]. ABE allows files to be encrypted under a access structured based on a set of attributes. This moves the problem of access management to the distribution of decryption keys for attributes. In a PHR these access structures allow a patient to gain fine-grained control over which HCPs are able to access which parts of his PHR.

3.5.1 *Primitives*

ABE comes in two forms, Key-Policy Attribute-Based Encryption (KP-ABE)[79] and Ciphertext-Policy Attribute-Based Encryption (CP-ABE)[80]. As the name implies, with KP-ABE the access structure is embedded in the nature of the keys. For CP-ABE this is embedded in the ciphertext itself.

One of the main challenges in ABE is that of user and attribute revocation. In real-world applications it is often desirable to be able to remove decryption rights from users. Works such as [81][82], add revocation through the combination of proxy re-encryption. Whenever a user's rights are revoked, the cloud storage re-encrypts any files encrypted under these attributes. The main drawback to this approach is that the re-encryption is costly and puts trust into the cloud storage. Furthermore, this approach is unpractical in a distributed storage setting. In [83] attribute revocation is realized by assigning a key to each attribute. In order to revoke an attribute, a key authority generates a new key for the attribute. The new key is send to all parties who can use it to update their decryption attributes and their attribute decryption keys. The work [84] relies on on the distribution of new keys for all the unaffected parties, whenever an attribute gets revoked. In [85] an improvement on this mechanism is proposd. If a user misses one of the key-updates, the user can still retrieve the correct decryption keys.

A big drawback of the ABE schemes mentioned so far, is that of the use of a centralized or trusted authority. Without such an authority, these schemes are not secure. In order to solve this several works attempt to distribute or remove this authority. In [86] a distributed pseudo-random function is used to generate keys and attributes, removing the need for a central authority. In [87] a threshold-based ABE scheme is proposed. Here attribute distribution is managed by several attribute authorities (AAs). Each AA is responsible for a subset of all attributes. When encypting a message, for each AA k, a number of required attributes d_k is picked. A party holding at least d_k attributes for for at least $t + 1$ authorities is able to decrypt.

3.5.2 *Applications*

In [88] an ABE-based architecture for a EHR is proposed. This system relies on smartcards, together with a security PIN, through which patients are able to authorize HCPs to access their data. A trusted key server sends decryption keys to authorized HCPs

Another example of the application of ABE for distributed cloud storage can be found in [89]. The scheme achieves fine-grained access control, anonymity, user authentication, and has distributed key management. Unfortunately, the access structure is visible to the cloud

provider, and there is no way of performing access monitoring without trusting the cloud provider.

In [90], and its extension [91], a PHR using ABE allowing for dynamic access policy modification, and on-demand attribute revocation is constructed. The scheme distributes the role of the attribute generating authority, using multi-authority ABE. In this setting each authority is responsible for a subset of the attributes, meaning that exploitation of a subset of attributes is still possible by a single authority.

3.6 SEARCHING OVER ENCRYPTED DATA

Storing the data in encrypted form means that performing search over the data becomes a challenge. Especially when SMDs are collecting data at every point of the day, meaning a lot of data will be gathered. In order to utilize this data stored in encrypted form, methods for searching over encrypted data can provide a solution.

3.6.1 *Primitives*

One way to provide search over encrypted data is by giving each file a limited set of keywords. Using a one-way function, such as a hash, these keywords are then hidden. The hash gets stored at the cloud provider, along with a pointer to the associated file. A search can then be performed by comparing the hashes of keywords. This provides a very basic search functionality, the keywords require an exact match, otherwise the query will give no result. In order to account for typos fuzzy search can be applied[92][93]. The edit distance between two words is the amount of operations required to turn one word into another. An operation can be the addition, removal or replacement of a character. In this setup a maximum edit distance between a searched keyword and a return keyword is defined. When a keyword is added to the database not only the keyword hash itself, but also all keyword hashes within the maximum edit distance are stored. When searching for a keyword a search is performed for the hash of the keyword itself and all keywords within a certain edit distance.

Given a collection of n encrypted files $C = (F_1, F2, ..., F_n)$ and p keywords $W = w_1, w_2, ..., w_p$. A cloud provider stores these encrypted files and keywords. The user wants to be able to search over these encrypted files based on the keywords. The edit distance $ed(w_1, w_2)$ measures the similarity between two keywords w_1 and w_2, and is defined as the amount of operations required to transform one of them into the other. These operations can be 1) Substitution: Replacing one character with another in the word; 2) Deletion: Removing a character from the word; 3) Insertion: Inserting a single character into the word. Let $S_{w,d}$ be the set of words w' satisfying $ed(w, w') \leqslant d$. For

fuzzy search the value of d needs to be predefined, and determines the maximum dissimilarity between a search query and the returned result. With this a search input can be defined as (w, k) with $k \leqslant d$ which will return either FID_w if present, otherwise all FID_{w_i} with $\mathrm{ed}(w, w_i) \leqslant d$. A naive approach to solve this problem would be to for every keyword w_i pre-compute all words w_i' with $\mathrm{ed}(w_i, w_i') \leqslant d$. Assuming an alphabet of 26 characters the amount of keywords required to be stored for keyword w with $d = 2$ will approximately become $2k^2 \cdot 26^2$. A search for (w, d) can then be performed by computing all words w' with $\mathrm{ed}(w, w') \leqslant d$ and sending this collection to the cloud server for comparison.

3.6.2 *Applications*

Though the method above demonstrates one method to achieve secure search, other works use different techniques. In [94], the secure evaluation of finite automata are used to search over DNA. Here a search query is formulated as a finite automata, after which its secure evaluation will return the answer to the search query.

In [95] a different approach is used to search over medical data. Here hierarchical predicate encryption is used in order to allow for multi-dimensional keyword search over encrypted data stored at a cloud provider. This allows search queries containing and excluding specific keywords, while keeping the contents of these queries hidden to the cloud provider.

3.7 CONCLUSION

In current literature we see many approaches for the secure handling of medical data. Each approache faces different challenges. For SMC, the challenge is that of translating existing algorithms into secure protocols, while keeping these protocols efficient. Other approaches deal with secure storage and retrieval of data. Here the challenge becomes storing data at untrusted cloud providers, granting access to only the right parties, and retrieving the right data.

SMC shows a lot of promise to provide medical data analysis, and automated monitoring of medical signals, while protecting both the party supplying the analysis and the data owner. However, in the current medical field, healthcare is in most cases still performed by humans looking at plaintext data. Because of this, a system where multiple parties can jointly manage a single PHR, where the patient remains in control, seems to have more utility in the near future.

The question which this chapter aims to answer is as follows,

What cryptographic methods are there to provide a secure, privacy-preserving PHR?

SMC shows the biggest potential for true private data handling, since private data never has to leave the machine. Though guaranteeing privacy, this approach in has many practical limitations. Storing all patient data locally would require a lot of storage space at the patient, and limite accessibility. Furthermore, all aspects of e-health can be translated into algorithms, let alone SMC algorithms. Because of this, a secure PHR is more critical for applicable e-health. In this PHR, the patient has insight into what data about him is being stored, and has full control over who is able to access this data. The data is stored in encrypted form, while SMC can enable parties to offer services for stored data, such as remote monitoring, diagnosis, and analytics. Cryptographic methods such as searching over encrypted data ensures the data can be utilized in a meaningful manner.

Regarding the secure data exchange for a PHR systems, we see several approaches. In order to asses the performance of these approaches, and the proposals for their application in e-health, we use common criteria, such as achieved security, complexity and efficiency. Furthermore, we define the following six properties in order to ensure compliance with the current regulations, functionalities present in the current Dutch e-health system, and to allow the secure integration of SMDs.

- *Confidentiality*, data can be stored securely, and with as little trust given to the cloud provider as possible.
- *Access control* is preferable controlled by the patient, the level of access control is also important.
- *Monitoring*, actions on the PHR are monitored and visible to the patient.
- *Anonymity*, the system provides anonymity over the data stored.
- *Trusted parties* need to be reduces. Trusted parties can be the cloud provider, but also HCPs or monitoring parties.
- *Integration of SMDs*, a solution which relies on human authentication would make integration of SMDs less practical.

We see that many approaches achieve multiple of the properties described above. However, few achieve all. Those who seem to achieve all or most of these properties, however, lack clear methods for the integration of SMDs, usually due to the reliance on smart cards [62][63][88].

Other solutions rely on the help of a trusted party. This trusted party is usually the key server[88], cloud provider[83], the HCP, or a separate (government) controlled entity[64]. Though others rely on a semi-trusted authority, these systems usually break when these authorities collaborate with the system users[23][65]. Though in general ABE-based methods show great potential for e-health systems, as they

allow for the construction of a PHR with fine-grained access control, none of the method meet our criteria.

Given the current and upcoming regulations (Sec. 2.2), we conclude that none of the proposed e-health systems is able to meet the requirements, without relying on the need for a centralized authority. This makes these systems insecure, if the central authority every is able to collaborate with other parties.

4

A DISTRIBUTED E-HEALTH SYSTEM

In this section we present Distributed Polymorphic Encryption and Pseudonymisation (Dist-PEP), our extension to Polymorphic Encryption and Pseudonymisation (PEP). We extend the work by removing the need for a centralized authority. We first explain PEP, how it works and its main security concerns. Afterwards, we explain our extension, resolving many of these concerns. The contributions explained in this chapter have been submitted as a paper to IEEE WIFS, which is currently under review. The full paper can be found in Appendix A.

4.1 POLYMORPHIC ENCRYPTION AND PSEUDONYMISATION

PEP[23] proposes a novel approach for the handling of medical data. Data gets encrypted at the end points and stored in encrypted form at a cloud provider. Afterwards, the data owner can decide which parties are allowed to retrieve the data. A patient will be known by different pseudonyms at every party. These pseudonyms are unlinkable across different parties. The main suggested application is that of a Personal Healthcare Record (PHR), where a patient can store data gathered by wearables at an untrusted cloud provider, and share it with HCPs.

PEP describes a setting where multiple parties are allowed to exchange data, but only with the help of a specialized party, called the Transcryptor (TR). This way, multiple parties, such as HCPs, patient, self-monitoring devices and cloud providers are able to exchange data(See Fig. 1). Since all data has to go through TR, he can perform access management and authorization. The TR is able to re-encrypt data, allowing multiple parties to decrypt the same data using their own private keys. Furthermore, TR replaces all identifiers with pseudonyms, which differ at every party, preventing linkage through identifiers.

4.1.1 *PEP operations*

The re-encryption and pseudonymisation process uses ElGamal[49] and its homomorphic properties. We deviate slightly from the notation used in PEP, in order to better explain our contributions. Let $\mathbb{G}$ be a group of prime order p, with generator g. Given a private key x and public key $y = g^x$, we encrypt a message M using $\text{EncEG}_y(M) := \langle g^r, y^r \cdot M \rangle$, using a random r. For decryption, we compute $\text{DecEG}_x(\langle b, c \rangle) := b^{x^{-1}} c$.

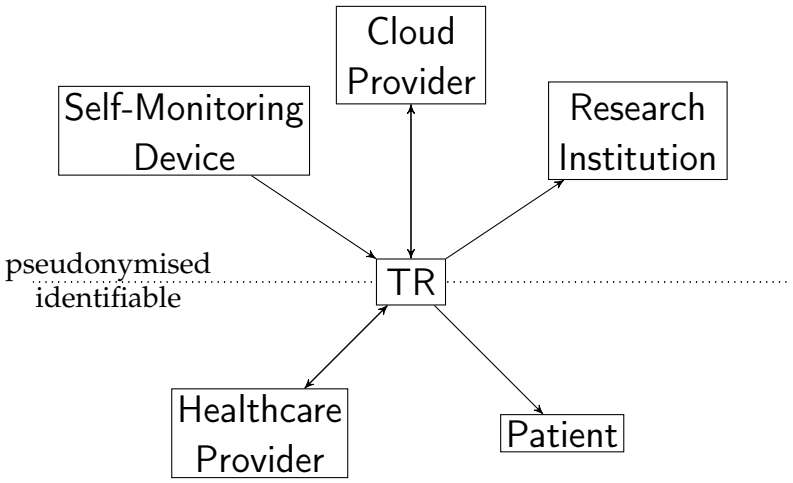

Figure 1: Data Flow in PEP

Table 1: Symbols and their meaning.

Symbol	Description
X, y	master key pair
X_i, y_i	key pair of i
k_i	key-factor for i
s_i	pseudonym-factor for i
ID_i	personal identifier of i
ID_i@j	pseudonym of i at j
EID_i	encrypted identifier of i
EID_i@j	encrypted identifier of i at j

Given an encryption of message M, $C = \text{EncEG}_y(M) = \langle b, c\rangle$, encrypted using public key $y = g^X$, PEP defines the following three homomorphic operations on C.

- *Randomize Elgamal* picks a random value r, and produces a randomized encryption of the same message,

$$\text{RandEG}_r(\langle b, c\rangle) \coloneqq \langle g^r \cdot b, y^r \cdot c\rangle \mod p. \tag{6}$$

- *Re-Key ElGamal* uses key-factor k, and produces an encryption which can be decrypted using different private key $X' = X \cdot k$,

$$\text{KeyEG}_k(\langle b, c\rangle) \coloneqq \langle b^{k^{-1}}, c\rangle \mod p, \tag{7}$$

- *Re-Shuffle ElGamal* performs a homomorphic exponentiation by s, creating a new encryption which will decrypt to M^s.

$$\text{ShuffleEG}_s(\langle b, c\rangle) \coloneqq \langle b^s, c^s\rangle \mod p. \tag{8}$$

4.1.2 *Storing and Retrieving Data*

The main operation in PEP is that of pseudonymisation. Given an identifier ID_i, every party j will instead learn a pseudonym, $ID_i@j = ID_i^{s_j}$ (The exponentiation of i's identifier ID_i, using j's pseudonym-factor s_j). Only TR knowns pseudonym-factor s_j and has to perform this exponentiation. In order to hide the identifiers and pseudonyms from TR, TR performs this exponentiation on an encyption of ID_i. We now explain how this pseudonymisation process can be used in order to store data gathered by a wearable at a Cloud Provider (CP), after which a HCP can retrieve it.

Setup, a key server generates a keypair with private key $\mathcal{X}$, and public key $\mathcal{Y} = g^{\mathcal{X}}$. For every party i, the key server generates a random key-factor k_i, and sends i their private key $\mathcal{X}_i = \mathcal{X} \cdot k_i$, and sends TR the key-factor k_i. For every participant i, TR generates a pseudonym-factor s_i. Every patient j is assigned a unique identifier, ID_j.

Storage. Patient P has a wearable device W, which wants to store some data D at a Cloud Provider CP. CP should not learn the contents of D, nor the identifier of P. W keeps an encryption of P's identifier as $EID_P = EncEG_{\mathcal{Y}}(ID_P)$. First W randomize the enccrypted identifier using random r, $EID'_P = RandEG_r(EID_P)$, and encrypts D, $E = EncEG_{\mathcal{Y}}(D)$. Next, W sends EID'_P and E to TR. TR re-keys and re-shuffles the encrypted identifier using the key-factor and pseudonym factor for CP, $EID_P@CP = KeyEG_{k_{CP}}(ShuffleEG_{s_{CP}}(EID'_P)$. TR sends $EID_P@CP$ and D to CP. CP can decrypt $EID_P@CP$ using his private key $\mathcal{X}_{CP}$ in order to obtain the pseudonym of P at CP, $ID_P@CP = DecEG_{\mathcal{X}_{CP}}(EID_P@CP) = ID_P^{k_{CP}}$. CP uses the pseudonym as an identifier to store E

Retrieval. CP holds an encryption E of data D, owned by patient P. Let A be a HCP authorized to retrieve data owned by P. A knows $EID_P = EncEG_{\mathcal{Y}}(ID_P)$. A randomizes the encrypted identifier using random r, $EID'_P = RandEG_r(EID_P)$ and sends it to TR. TR re-keys and re-shuffles using the key-factor and pseudonym-factor for CP, $EID_P@CP = KeyEG_{k_{CP}}(ShuffleEG_{s_{CP}}(EID'_P)$ and sends the result to CP. CP decrypts to obtain the pseudonym $ID_P@CP$ and use it as an index to retrieve E. CP randomizes E as E', and sends E' to TR. TR re-encrypts E' for A, $C = KeyEG_{k_A}(E')$ and sends the result to A. A can decrypt using his private key in order to obtain D, $DecEG_{\mathcal{X}_A}(C) = D$.

4.1.2.1 *Limitation of PEP*

Through the pseudonymization process, participants have to communicate through the TR. This creates a central party, who can perform access management, logging and monitoring. However, if TR ever chooses to collaborate with any other party, a lot of data will be

leaked. In this section we discuss this and several other limitations of PEP.

- Assumes the existence of a fully-trusted access manager.
- The key server holds the master private key $\mathcal{X}$. If key server collaborates with the storage facility, all data and all polymorphic pseudonyms can be decrypted.
- TR holds secrets k_i and S_i for each party i. Each i A holds $\mathcal{X}_i = \mathcal{X} \cdot k_i$. If TR collaborates with any i, they are able to learn master private key $\mathcal{X}$, creating the same security threat as compromising the key server. Furthermore they are able to compute any other private key for party j, $\mathcal{X}_j = \mathcal{X}_i \cdot k_j/k_i$, and are able to learn personal identifier ID_j from any pseudonym of j.

In order to reduce the control of the TR, PEP suggests methods in which TR has to collaborate with an access manager in order to prevent this, however this just moves the trust to the access manager, leaving many of the same security risks. Furthermore, they proposes the use of a Hardware Security Module (HSM). These HSMs manage cryptographic keys, key-factors and pseudonym-factors. All PEP operations are performed inside these HSMs. Though this prevents direct leakage of these secret values, TR is still able to transform any data or pseudonym, such that it can be decrypted by any participant.

Besides security threats, the construction of PEP can only be scaled vertically. Keeping the context of wearables in mind, we can only expect the amount of data which the TR needs to process to increase. Furthermore whenever TR goes offline, no exchange of data can take place.

4.2 DISTRIBUTED POLYMORPHIC ENCRYPTION AND PSEUDONYMISATION

PEP relies on a single TR in order to re-encrypt data and pseudonymise identifiers. We instead propose Dist-PEP, a setup where these operations are performed by a group of n TRs. Within this group, a quorum of t TRs need to collaborate in order to re-encrypt and pseudonymise. Furthermore, we remove the need for a trusted key server, by letting TRs collaborate, in order to generate private keys, key-factors and pseudonym-factors. The result is a scheme which is more scalable, when the demand for bandwidth increases, we can increase n. Furthermore we remove the need for a central semi-trusted authority (TR), and the huge security risks when this authority chooses to collaborate with any other party (Leakage of all stored data, identifiers, and pseudonyms). We achieve this using Shamir Secret Sharing (See Sec. 3.1.1.4).

4.2.1 *Dist-PEP Operations*

We demonstrate how the homomorphic operations defined in PEP can be performed in a distributed manner using Secret Sharing. Given an encryption of message M, $C = \text{EncEG}_y(M) \coloneqq \langle b, c\rangle$, and quorum π of t TRs. Send C to every member i of π. Every i 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 the *Combine ElGamal* operation, $\mathit{CombEG}_\pi(P) \coloneqq \langle \prod_{i\in\pi} b_i, \prod_{i\in\pi} c_i\rangle$.

Given a secret sharing $[s]$ and $[k]$, a partial results p_i can be computed as follows.

1. *Partial Randomize ElGamal*, using a random number r, $\text{PartRandEG}_r(\langle b, c\rangle) \coloneqq \langle b^{t^{-1}} g^r, c^{t^{-1}} y^r\rangle$.
2. *Partial Re-Key ElGamal*, using $[k]$, pre-compute a secret sharing of $[k^{-1}]$. $\text{PartKeyEG}_{[k]}(\langle b, c\rangle) \coloneqq \langle b^{[k^{-1}]^\pi_i}, c^{t^{-1}}\rangle$.
3. *Partial Re-Shuffle ElGamal*, $\text{PartShuffleEG}_{[s]}(\langle b, c\rangle) \coloneqq \langle b^{[s]^\pi_i}, c^{[s]^\pi_i}\rangle$.
4. *Partial Re-KeyShuffle ElGamal*, using $[k]$ and $[s]$, pre-compute a secret sharing of $[sk^{-1}]$, $\text{PartKeyShuffleEG}_{[k],[s]}(\langle b, c\rangle) \coloneqq \langle b^{[sk^{-1}]^\pi_i}, c^{[s]^\pi_i}\rangle$.

Note that the pre-computations of new secret shares, such as computing $[sk^{-1}]$ from $[k]$ and $[s]$ only needs to be done once. Afterwards the secret shares of $[sk^{-1}]$ are stored, speeding up the computation in the future.

4.2.2 *Removing the Trusted Key Server*

In the distributed setting, we can remove the need for a trusted key server by distributing this role among the TRs. For this, the master secret key $\mathcal{X}$ is generated as a secret sharing of random number by the TRs, $[\mathcal{X}]$. This secret sharing is done using threshold t', which can be higher than the secret sharing of pseudonym-factors and key-factors, increasing the amount of malicious TRs which would have to collaborate in order to retrieve $\mathcal{X}$.

Let i be a party joining the system. In order to generate a key-factor, pseudonym-factor and a private key for i, a quorum π of at least $2t' - 1$ TRs collaborate in order to generate secret sharings of random values for key-factor $[k_i]$ and pseudonym-factor $[s_i]$. Each $j \in \pi$ compute $[\mathcal{X} \cdot k_i]^\pi_j$ and sends this to i. i can computes his private key as $\mathcal{X}_i = \sum_{j\in\pi} [\mathcal{X} \cdot k_i]^\pi_j = \mathcal{X} \cdot k_i$.

4.2.3 *Adding a new transcryptor*

After setup, it is possible to add or replace a TR. Let TR_k be a new TR joining the setup. For this the existing TR simply collaborate in order

Table 2: Computational complexity Dist-PEP operations.

Operation	**Multiplication**	**Exponentiation**	**Overall Complexity**
CombEG	$2t$	0	$O(t)$
PartRandEG	$2t$	$4t$	$O(t)$
PartKeyEG	0	$2t$	$O(t)$
PartShuffleEG	0	$2t$	$O(t)$
PartKeyShuffleEG	0	$2t$	$O(t)$

to give the new TR secret shares of the values currently shared among the TRs.

4.3 ANALYSIS

In this section we shall present a security sketch of our scheme. This is followed by an analysis of its complexity. Finally using an implementation we test its performance.

4.3.1 *Security*

Shamir Secret Sharing guarantees that for a t out of n secret sharing, nothing can be learned about the shared secret with less than t shares. This means with $t-1$ collaborating TRs can not learn anything about any k_j or s_j. $t'-1$ TRs are unable to determine anything about $\mathcal{X}$. The unlinkability of randomized encryptions and randomized encrypted pseudonyms relies on the Decisional Diffie-Hellman problem: Given $g, g^a, g^b, g^c \in \mathbb{G}$, it should be infeasible to determine if $a \cdot b = c$.

4.3.2 *Complexity*

We perform complexity analysis of our scheme based on the amount of multiplications and exponentiation performed. The CombEG operation is analyzed for one participant, while the PartRandEG, PartKeyEG, PartShuffleEG and PartyKeyShuffleEG operations are analyzed as if they are performed by t TRs (See Table 2). All operations show a complexity of $O(t)$.

All PEP operations in the original scheme have complexity of $O(1)$. Note that for completeness, we provide an analysis for the PartRandEG operation, however, this operation is not used by our scheme.

For any Dist-PEP operation the message complexity analysis is as follows. A participant needs to send a encrypted pseudonym and encryption of data to t TRs. After performing their partial computation, t TRs send their partial result to a single receiver. This gives us a tight bound of $\theta(2t)$ messages.

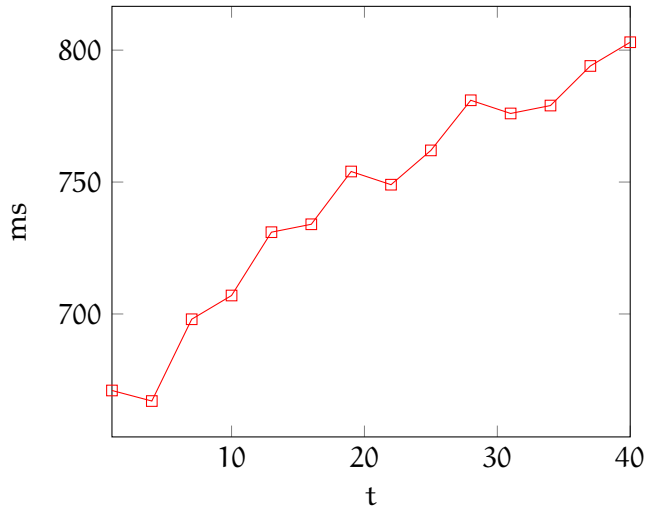

Figure 2: PartKeyShuffleEG computation time

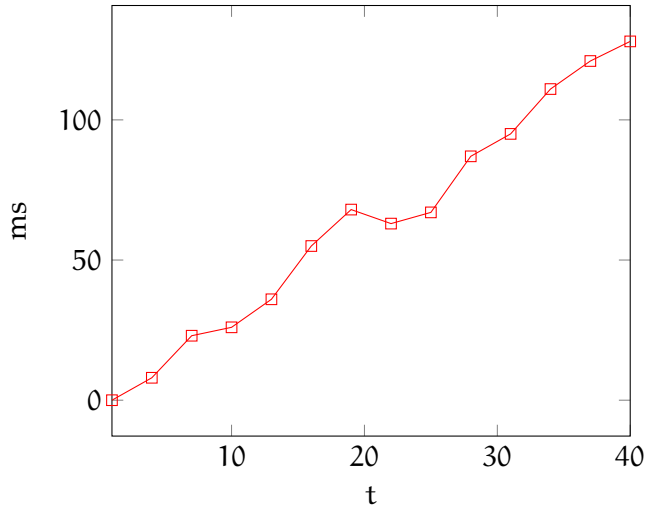

Figure 3: CombEG computation time

4.3.3 *Performance*

We have implemented the protocol using C++ and the GMP library[1]. We test the performance on a machine running Windows 10.0, with a Intel Core i5-7200 running at 2.50GHHz. We analyze the computation time for performing t PartKeyShuffleEG operations, and combining the partial results using CombEG, for different values of t. The PartKeyShuffleEG computation time is the average time spend by a single TR. The resuls can be seen in Fig. 7 and Fig. 8.

We plot the results in Fig. 2 and 3. As expected both methods demonstrate a linear complexity of $O(t)$.

1 See https://gmplib.org/.

4.4 CONCLUSION AND DISCUSSION

In this chapter we propose an approach to controlling data access in a distributed manner. Through pseudonymisation, we provide anonymity, while enforcing that any exchange of data has to pass through a quorum of transcryptors. This allows checking of access rights and monitoring. Furthermore, this is done in a distributed manner, using Shamir Secret Sharing, increasing scalability and security. We further improve on previous solutions, by removing the need for a key server. Through complexity analysis, and by measuring the performance of an actual implementation, we demonstrate the efficiency of our protocol.

Though the work in this chapter shows an improvement over PEP in terms of privacy and security, some of its flaws are still present. Most notable the reliance on a trusted access manager. In the current approach the access manager dictates in which data transactions the group of TRs will assist.

5 PATIENT CONTROLLED ACCESS MANAGEMENT

In the previous chapter we present Dist-PEP. The scheme removes the need for a centralized trusted party, however, still relies on a fully trusted access manager. In this section we present Polymorphic Access Management (PACMAN). PACMAN further improves Dist-PEP, by embedding access management into the encryption. The result is an e-health system where multiple parties can submit data to a single patient's PHR. The patient has control over which parties are able to access which parts of his PHR. This exchange of data is managed by a group of parties. A certain threshold of these parties need to collaborate in order to facilitate in the exchange of data.

We present this contribution using a paper, which is planned to be submitted to IEEE TIFS. The contents of this paper should provide an answer to our third research question,

How can the patient stay in control of his medical data?

The paper is structured as follows. It starts with an introduction explaining the current issues with e-health, and the integration of self-monitoring devices. Furthermore, the introduction contains a statement about the significance of our contribution. Afterwards, it explains our system and security model, and covers the preliminaries needed for our construction. The preliminaries contain a summary Dist-PEP, the subject of the previous chapter. Afterwards we PACMAN, our contribution. We compare our construction to previous constructions, both in terms of complexity and security. Next, we show some further improvements to the scheme, which in order to provide a fair comparison have been left outside of the analysis. Finally, we explain how our construction could be implementated in order to create a PHR.

5.1 POLYMORPHIC ACCESS MANAGEMENT FOR PERSONAL HEALTHCARE RECORDS

In recent years the market for wearable devices, such as smartwatches and fitness trackers, has emerged. This market is expected to grow, from 84 million units sold in 2015, to an expected 245 million units in 2019[3]. The majority of these devices are currently equipped with sensors such as GPS, step counters, and heart monitors. However, more specialized devices are able to perform more advanced measurements, such as blood sugar levels, oxygen levels, blood pressure, and ECGs[96]. Most manufacturers of these devices, provide the user

with the functionality of syncing these measurements in the cloud, and for some devices this is the only way of extracting measurements. Currently these devices are mainly used by consumers for self-monitoring: keeping track of ones personal fitness. However, as these devices are becoming more advanced, their application is moving towards medical purposes.

The idea of a personal healthcare record (PHR), a digital healthcare record in which the patient can create, manage and control his personal medical data in one place, has been envisioned for quite some time. The PHR would allow multiple healthcare providers (HCPs) to contribute to the same record, while the patient controls who is allowed to access which parts of the record. Such a PHR would furthermore allow the patient to contribute to his record using self-monitoring devices. A good demonstration of the potential of the PHR, is that of multiple HCPs prescribing medication to the same patient. In this situation it is important for the HCPs to be aware of the medication prescribed by the other party, to avoid unexpected interactions between the medication. Instant access to parts of the patients PHR would solve this problem. With the incorporation of self-monitoring data, the potential becomes even more promising. In this setup data is gathered by (wearable) devices, and stored in the cloud at a storage facility. The patient grants the HCP access to measurements relevant for his treatment, replacing physical check-ups by remote monitoring by the HCP. This shows promise in saving costs, by reducing both the amount of hospital visits required, as well as reducing the work performed by the HCP. Furthermore, the HCP can rely on measurements taken in a natural setting, over a longer period of time, possibly increasing the quality of the care provided.

The benefits of a PHR should be clear, however, given the sensitivity of the data involved, big privacy and security challenges arise. A patient wants a HCP to be able to only access information relevant for providing the healthcare. The current state of self-monitoring is that the data is stored in plaintext at the manufacturer, giving the manufacturer full control over the data. Manufacturers use the gathered data for secondary purposes, such as targeted advertisements, sharing aggregates with research institutions, and using it for analytics about the usage of their products.

To prevent the exploitation of the PHR, one approach is to prevent the sharing of data altogether. Instead, such schemes apply secure multi-party computation, in order to form a diagnosis over the measurements. Examples are the use of binary decision trees[61] to perform remote diagnostics on encrypted data. More advanced examples are the use of private linear branching programs[57] and private neural networks[60], in order obtain a classification of ECG signals, where both the classification algorithm and the ECG measurements stay private.

Though the current state of diagnostic algorithms show promise, in the current state of the medical field, healthcare is generally still conducted by humans. This requires the exchange of plaintext data, creating the different challenge of providing access management and data privacy, while storing the data at an untrusted storage provider. This task is further complicate, given that multiple HCPs ans self-monitoring devices need to contribute to a single PHR.

The use of polymorphic encryption and pseudonymisation (PEP) has been proposed[23], in order to reduce the effectiveness of combining data from different parties. This allows multiple parties to contribute and exchange data in a PHR, while learning little about the identity of the PHR owner. In PEP an identifier is replaced by pseudonyms. These pseudonyms look different at every party, preventing linkage. Generating the pseudonyms from the identifier is performed by a centralized party. One of the main drawbacks of this method is the big role of this central party. Through collaboration this central party is able to find the relationship between all identifiers and pseudonyms, and even decrypt all data. Furthermore the method assumes proper access control.

Attribute-Based Encryption (ABE) has been researched, to solve the access management challenge for PHRs[97], and cloud storage in general[84]. ABE allows access management based on an access structure. Two main types of ABE exists, Key-Policy ABE (KP-ABE)[79] and Ciphertext-Policy ABE (CP-ABE)[80]. The difference between the two methods is that in the first, the access structure is embedded in the key, while in the second the access structure is part of the encryption. This means that control over the access structure in the first lies with the key authority, while the second lies with the encrypting party. Furthermore, both forms require some form of trusted authority to handle out keys and attributes. Furthermore, the access structure needs to be known to decrypt an ABE encryption. Because of this, the access structure is public, revealing information about the content of the encryption.

In this work, we propose two novel constructions, providing access management with application for PHRs. The first construction combines pseudonymisation with CP-ABE. By embedding the access structure into the encryption, we remove the need for a trusted access manager. Only a party owning the attributes to fulfill the access structures is able to decrypt. The exchange of such ciphers can only take place with the help of an intermediate party, which means any action on the PHR can be monitored, and is visible to the PHR owner. Through pseudonymisation we limit data leakage through the access structure.

In the second construction, we extend the first, by distributed the role of the intermediat party. The result is a group of intermediate parties. Within this group a quorum of a certain threshold has to be

reached in order to perform any action on the PHR. This greatly reduces the security risks of malicious parties choosing to collaborate. Furthermore, the intermediate parties are able to remove the need for a trusted party all together, by distributing key generation and the distribution of attributes. In addition, the construction allows for easy key-revocation with minimal computation cost, and the implementation of time-based access control.

To the best of our knowledge, this is the first construction achieving the combination of the following:

1. Multiple parties contributing to a dataset with a single data owner
2. Fine-grained access control for the data owner
3. Decentralized storage
4. No centralized (semi-)trusted authority for key and attribute management
5. Resistant to collusion of multiple parties
6. Unlinkability of data owner across different parties
7. Efficient key revocation, without re-encryption of stored encryptions
8. Time-based access control

The rest of this work is organized as follows. In Sec. 5.2 we explain the preliminaries. Next in Sec. 5.3 we explain our main contribution: combining attribute based encryption with pseudonomyzation. Next in Sec. 5.4 we expand this idea to work in a distributed setting. Afterwards in Sec. 5.5 an analysis of the construction is performed in terms of complexity and security. In Sec. 5.8 we explore an application of Dist-PACMAN for storing medical data.

5.2 PRELIMINARIES

5.2.1 *System Models*

For the first construction, we assume the following parties. Participants, these are the patient or data owner, cloud providers (CPs) and healthcare providers (HCPs). The patient, or data owner of the PHR. The data owner adds data to the PHR through the use of self-monitoring devices. Furthermore, the data owner needs to have access control over his PHR. Any action on the PHR needs to be visible to the data owner. The CPs provide storage for the PHR, but are not able to view the content of any stored data. Each HCP is allowed

Table 3: List of Definitions

Name	Symbol	Description
number of transcryptors	n	-
threshold of transcryptors	t	-
quorum of t transcryptors	π	$\lvert\pi\rvert = t$
lagrange coefficient	$\Delta_i^\pi(x)$	$\Delta_i^\pi(x) = \prod_{j\in\pi, j\neq i} \frac{x-j}{i-j}$
secret sharing of S	$[S]$	-
partial solution of S	$[S]_i^\pi$	$\sum_{i\in\pi}[S]_i^\pi = S$
bilinear map operation	$e(x, y)$	$e(a^u, b^v) = e(a, b)^{uv}$
ElGamal private key	$\mathcal{X}$	$\mathcal{X} \in_R \mathbb{Z}_p$
ElGamal public key	$\mathcal{Y}$	$\mathcal{Y} = g^{\mathcal{X}}$
key-factor of A	k_A	$k_A \in_R \mathbb{Z}_p$
pseudonym-factor of A	s_A	$s_A \in_R \mathbb{Z}_p$
ElGamal private key of A	$\mathcal{X}_A$	$\mathcal{X}_A = \mathcal{X} \cdot k_A$
ElGamal public key of A	$\mathcal{Y}_A$	$\mathcal{Y}_A = \mathcal{Y}^{k_A}$
access tree	$\mathcal{T}$	-
pseudonym of $\mathcal{T}$ at A	$\mathcal{T}@A$	-
leafs of access tree	Y	-
threshold of node x	t_x	-
children of node x	S_x	-
encrypted access tree	ET	$\text{ET} = \text{EncTree}_{\mathcal{Y}}(\mathcal{T})$
number of nodes in access tree	η	$\eta = \lvert\mathcal{T}\rvert$
number of leafs in access tree	ℓ	-

access to a subset of the PHR, controlled by the data owner. Furthermore, HCPs are able to contribute data to the PHR. If the data owner grants consent, research institutions are able to obtain anonymised data from the patient's PHR. Besides the participants we have an intermediate party, called the transcryptor (TR). TR is an independent party, assisting in the exchange of PHR data. TR should learn nothing about the data it is processing, nor the data owner. How data flows between these different parties can be seen in Fig. 4.

In the second construction, the role of the TR is distributed. The result is that we have multiple TRs. Data instead flows through a quorum within this group of TRs.

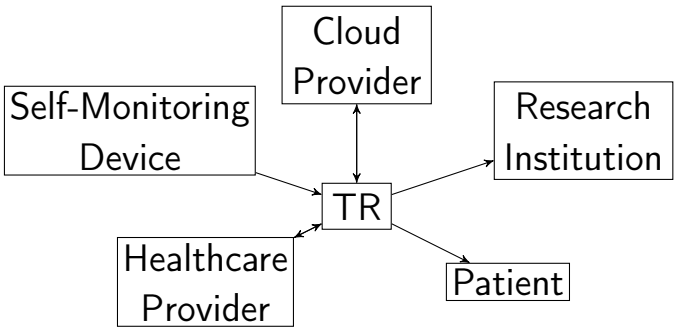

Figure 4: Data Flow in System Model

5.2.2 *Security Models*

We assume all parties are semitrusted, i.e. they operate in an honest but curious manner. This menas each party will not deviate from the protocol, however, they will try to gain more information then the protocol prescribes. We assume that communication between all parties is performed over a secure channel.

In addition, in the second construction, we assume that at most a threshold of $t-1$ TRs decide to collaborate in order to learn as much secret information as possible.

5.2.3 *Cryptographic building blocks*

In this section we explain cryptographic building blocks, used in order to form our constructions. In Sec. 5.2.3.1, we start with the notion of Secret Sharing. Secret Sharing allows the distribution of a secret among a group of parties, after which a subgroup can retrieve the secret. Secret Sharing forms the basis of Attribute-Based Encryption (ABE), which will be introduced in Sec. 5.2.3.2, and is the main method for distributing the role of the TR in the second construction. In Sec. 5.2.3.3, we explain Polymorphic Encryption and Pseudonymisation (PEP), which uses a central party to replace identifiers by differ-

ent pseudonyms, which are unique at every party, preventing linkage. PEP is used in order to pseudonymise the access structures used in ABE. Finaly, in Sec. 5.2.3.4, we explain distributed PEP, a form of PEP where the role of the centralized party is distributed.

5.2.3.1 *Secret Sharing*

Using a t out of n secret sharing scheme a secret value S can be divided into n shares. At least $t \leqslant n$ shares are needed in order to retrieve secret value S[50]. Let $[S]$ denote such a secret sharing of S, $[S]$ can be constructed as follows. Pick a random polynomial f of degree $t-1$, such that $f(0) = S$. For every share i, let x_i be a unique public value. For every share i compute $y_i = f(x_i)$ as its secret value. Given a quorum π of shares, with $|\pi| = t$, we can compute S as follows. First we define the formula for the Lagrange coefficient as

$$\Delta_i^{\pi}(x) = \prod_{j \in \pi, j \neq i} \frac{x-j}{i-j}. \tag{9}$$

Now we compute t partial solutions as

$$[S]_i^{\pi} = y_i \Delta_i^{\pi}(0), \tag{10}$$

which add up to S,

$$\sum_{i \in \pi} [S]_i^{\pi} = S. \tag{11}$$

Using two secret sharings $[A]$ and $[B]$, it is possible to perform multi-party computations, of which the outcome will be of the form of a new secret sharing. This work relies on the computation of addition ($[A+B]$)[98], multiplication ($[A \cdot B]$)[51], and modular inverse ($[A^{-1} \mod p]$)[52] of such secret shares. Important to note, is the effect these operations have on the thresholds of the secret sharings. Let t_A and t_B be the threshold of secret sharing $[S]$ and $[B]$ respectively. For addition, the size of the quorum required for operation is equal to $\max(t_a, t_b)$. For multiplication and exponentiation, this size is $t_a + t_b - 1$. Furthermore, a quorum of size t, can generate a secret sharing of a random value with any threshold smaller than t, where the generated secret stays hidden to all parties. As demonstrated in [55][53], public values can be exponentiated by a shared secret, where only a single receiving party learns the result of the computation.

5.2.3.2 *Ciphertext-Policy Attribute-Based Encryption*

In Ciphertext-Policy Attribute-Based Encryption (CP-ABE) access is managed using attributes and embedded in the encryption[80]. This in contrast to Key-Policy ABE, where the access structure is embedded in the construction of decryption keys[79]. Using these attributes

an access structure can be defined in the form of a tree. Every non leaf node holds a threshold value, and every leaf node holds an attribute. Only a party holding the right attributes to satisfy the tree is able to satisfy the access structure. We first explain how these access trees are constructed. Then we explain the notion of bilinear maps. Finally we explain how access trees and bilinear maps can be used to form the CP-ABE scheme.

Access trees. Let $\mathcal{T}$ be a tree constructed as follows. Every non-leaf node x holds its set of children S_x, and a threshold value t_x. Let $parent(x)$ denote the parent node of x, and $index(x)$ denote the index of x within $S_{parent(x)}$, ranging from 1 to $|S_{parent(x)}|$. Every leaf node has $t_x = 1$ and an associated attribute att_x.

Satisfying an access tree. Let $\mathcal{T}$ be an access tree and γ a set of attributes. Let $\mathcal{T}_x$ be a subtree of $\mathcal{T}$ rooted at node x. If γ satisfies access tree $\mathcal{T}_x$ this is denoted as $\mathcal{T}_x(\gamma) = 1$. If x is a leaf-node then $\mathcal{T}_x(\gamma) = 1$ if $att_x \in \gamma$. Otherwise $\mathcal{T}_x(\gamma) = 1$ if $\sum_{z \in S_x} \mathcal{T}_z(\gamma) \geqslant t_x$. Satisfying $\mathcal{T}$ means satisfying the subtree rooted in the rootnode of $\mathcal{T}$.

Bilinear Maps. Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be multiplicative groups of prime order p, with g being a generator of $\mathbb{G}_0$. Let e be a bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. e has the following properties:

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

Group $\mathbb{G}_0$ is considered a bilinear group if the group operation in $\mathbb{G}_0$ and the bilinear map e can be computed efficiently. Using generator g we find that $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$, showing that map e is symmetric.

Attribute Based Encryption.

- *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 generate secret key $SK = \langle \beta, g^\alpha \rangle$. Publish public key

$$PK = \langle \mathbb{G}_0, g, h = g^\beta, f = g^{\beta^{-1}}, e(g, g)^\alpha \rangle. \tag{12}$$

- *Key Generation*, takes set of attributes S and outputs a decryption key for the attributes in S. Pick random $r \in_R \mathbb{Z}_p$ and for every $j \in S$ pick $r_j \in_R \mathbb{Z}_p$.

$$\begin{aligned} KeyGenABE_{MK}(S) = \langle D = f^{(\alpha+r)}, \\ \forall j \in S : D_j = g^r \cdot H(j)^{r_j}, D'_j = g^{r_j} \rangle \end{aligned} \tag{13}$$

- *Encrypt ABE*, encrypts M under access tree $\mathcal{T}$. Pick a random number $w \in_R \mathbb{Z}_p$. For every node x in $\mathcal{T}$, generate a random

polynomial q_x with $q_x(0) = q_{parent(x)}(index(x))$. Root node r has no parent and instead has $q_r(0) = w$. Let Y be the set of leaf nodes in $\mathcal{T}$. For every leaf node $y \in Y$ compute $C_y = g^{q_y(0)}, C'_y = H(att_y)^{q_y(0)}$. Return resulting ciphertext as

$$\begin{aligned} \text{EncABE}_{PK}(M, \mathcal{T}) = \langle \mathcal{T}, \tilde{C} = Me(g, g,)^{\alpha w}, C = h^w, \\ \forall y \in Y : C_y = g^{q_y(0)}, \\ C'_y = H(att_y)^{q_y(0)} \rangle. \end{aligned} \tag{14}$$

- *Decrypt ABE*. Decryption is defined recursively. Function Decryptnode(CT, SK, x) takes a ciphertext $CT = \langle \mathcal{T}, \tilde{C}, C, \forall y \in Y : C_y, C'_y \rangle$, private key SK associated with set S of attributes and a node x. If x is a leaf node and $i \in S$, then let $i = att_x$,

$$\begin{aligned} \text{DecyptNode}_{SK}(CT, x) &= \frac{e(D_i, C_x)}{e(D'_i, D'_x)} \\ &= e(g, g)^{r q_x(0)}. \end{aligned} \tag{15}$$

If $i \notin S$, DecryptNode returns $\perp$. If x is not a leaf node, for every $z \in S_x$, compute $\text{DecryptNode}_{SK}(CT, z)$ and store the result as F_z. If for at least t_x elements in S_x, $F_x \neq \perp$, let $S'_x = \{index(z) : z \in S_x\}$ and compute

$$\text{DecryptNode}_{SK}(CT, x) = \prod_{z \in S_x} F_z^{\Delta_i^{S'_x}(0)} = e(g, g)^{r q_x(0)}. \tag{16}$$

Otherwise DecryptNode returns $\perp$.

In order to decrypt CT, decrypt root node r, $A = \text{DecryptNode}(CT, SK, r)$. Afterwards compute $\frac{\tilde{C}}{e(C,D)A^{-1}}$ to obtain the original message M.

5.2.3.3 *Polymorphic Encryption and Pseudonymisation*

Polymorhpic Encryption and Pseudonymisation (PEP)[23] provides data privacy by storing data in encrypted form. Furthermore, an individual will be known by a different identifier (called a pseudonym) at different parties. Let ID_A denote the identifier of A. At any other party B, A will be known by a related identifier, called a pseudonym $ID_A@B = ID_A^{k_B}$, with $k_B \in_R \mathbb{Z}_p$. The act of computing $ID_A@B$ from ID_A is performed by a third party called the Transcryptor (TR). This operation is performed on an encryption of ID_A, using a homomorphic cryptosystem. This ensures TR learns nothing about the values of ID_A and $ID_A@B$. This idea is demonstrated in Fig. 5. Party A encrypts his pseudonym $ID_A@A$ using his public key y_A. The encryption E is send to TR, who performs an operation, obtaining E'. TR sends E' to B, who can decrypt using his private key x_B, obtaining his local pseudonym for A, $ID_A@B$.

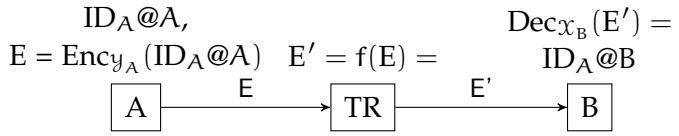

Figure 5: PEP setup.

PEP Operations. In order to securely compute pseudonyms from identifiers and other pseudonyms, PEP makes use of operations, exploiting the homomorphic peroperties of ElGamal[49]. We first explain how these operations work. Afterwards, we show how they can be used in practice to exchange pseudonyms and data in a privacy-preserving manner.

Let $\mathbb{G}$ be a group of prime order p, generator g. The private key x is a random number in $\mathbb{Z}_p$, from which the public key $y = g^x$ is computed. For encryption, pick a random $r \in \mathbb{Z}_p$ and perform $\text{EncEG}_y(M) \coloneqq \langle g^r, My^r \rangle$. The decryption operation is $\text{DecEG}_x(\langle b, c \rangle) \coloneqq cb^{-x}$.

PEP defines four operations on ElGamal encryptions. We modify the notation to align with our construction. Let $C = \text{EncEG}_y(M) = \langle b, c \rangle$, an encryption of M under public key $y = g^x$.

1. *Randomize ElGamal* takes encryption C and a random number r', and produces a randomized encryption of the same message, $\text{RandEG}_{r'}(\langle b, c \rangle) \coloneqq \langle bg^{r'}, cy^{r'} \rangle$.

2. *Re-Key ElGamal* takes k and re-encrypts C, so that it can only be decrypted using a different secret key of the form $x' = x \cdot k$. $\textit{KeyEG}_k(\langle b, c \rangle) \coloneqq \langle b^{k^{-1}}, c \rangle$.

3. *Re-Shuffle ElGamal* takes takes encryption C and value s, and creates an encryption of M^s, $\text{ShuffleEG}_s(\langle b, c \rangle) \coloneqq \langle b^s, c^s \rangle$.

4. *Re-KeyShuffle ElGamal* combines the Key and Shuffle ElGamal operations, $\text{KeyShuffleEG}_{k,s}(\langle b, c \rangle) \coloneqq \langle b^{sk^{-1}}, c^s \rangle$.

Storage and Retrieval, A wants to store data D in encrypted form at Storage Facility CP, after which B is allowed to retrieve and decrypt it. A holds an encryption of his pseudonym, $\text{EID}_A@A = \mathcal{EM}_{y_A}(\text{ID}_A@A)$. First, A Re-Randomizes $\text{EID}_A@A$, $\text{EID}_A@A' = \mathcal{RM}_r(\text{EID}_A@A)$. A encrypts the data as $E = \mathcal{EM}_y(D)$ and sends $\text{EID}_A@A'$ and E to TR. TR Re-KeyShuffles $\text{EID}_A@A'$ using $k_{A\to CP} = k_{CP}/k_A$ and $s_{A\to CP} = s_{CP}/s_A$, $\text{EID}_A@CP = \mathcal{KSM}_{k_{A\to CP}, s_{A\to CP}}(\text{EID}_A@A')$, and sends the result together with E to CP. CP can decrypt $\text{EID}_A@CP$ to obtain the pseudonym of A at CP, $\text{ID}_A@CP = \mathcal{DM}_{x_{CP}}(\text{EID}_A@CP) = \text{ID}_A^{s_{CP}}$. CP can not decrypt E and uses $\text{ID}_A@CP$ as an index to store E.

B holds $\text{ID}_A@B$, the pseudonym of A at B. B encrypts this pseudonym as $\text{EID}_A@B = \mathcal{EM}_{y_B}(\text{ID}_A@B)$ and sends it to TR. TR Re-KeyShuffles using $k_{B\to CP} = k_{CP}/k_B$ and $s_{B\to CP} = s_{CP}/s_B$, obtaining $\text{EID}_A@CP =$

$\mathcal{RKS}_{k_{B\to CP},s_{B\to CP}}(EID_A@B)$. TR sends $EID_A@CP$ to CP, who can decrypt to $ID_A@CP$, which he uses to retrieve E. CP sends E to TR who Re-Keys it using k_B, $E' = \mathcal{RK}_{k_B}(E)$ and sends E' to B. B decrypt E' to obtain D.

5.2.3.4 *Distributed Polymorphic Encryption And Pseudonymisation*

In Distributed Polymorphic Encryption and Pseudonyization (Dist-PEP), the role of TR in PEP (Sec. 5.2.3.3) is distributed among n parties using Secret Sharing. The resulting setup can be seen in Fig. 6. Here computation of different pseudonyms is performed in a distributed manner.

This distribution, makes use of the fact that all homomorphic operations in PEP, consist of exponentiations. Distributed PEP performs these exponentiations in a distributed manner, using Secret Sharing (Sec. 5.2.3.1). Let $[S]$ be a secret sharing of S with threshold t. Given a number x and a quorum π of t share holders, we can compute x^S in a distributed manner, without revealing S, $[S]$, or any of its shares. Furthermore, only one party learns x^S. For this, every $i \in \pi$ computes partial result $p_i = x^{[S]_i^\pi}$, and sends this to the receiving party. The receiving party computes $\prod i \in \pi p_i = g^S$.

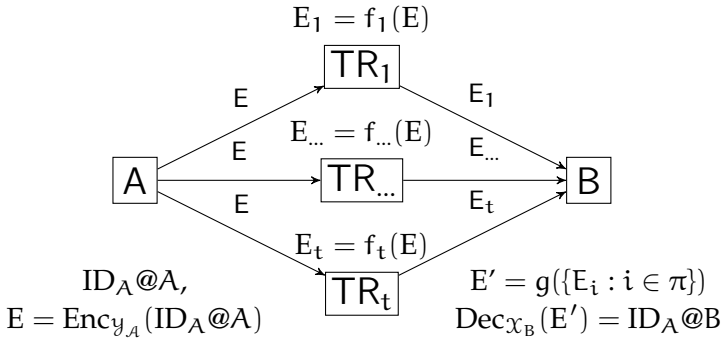

Figure 6: Distributed PEP setup.

Dist-PEP Operations. Using the idea of distributed exponentiation, the homomorphic operations of PEP can be performed in a distributed manner. Given an ElGamal encryption of message M, $C = EncEG_y(M) := \langle b, c\rangle$, and quorum π of size t. Send C to every member i of π. Every i 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 the *Combine ElGamal* operation, $CombEG_\pi(P) := \langle \prod_{i\in\pi} b_i, \prod_{i\in\pi} c_i\rangle$.

Given a secret sharing $[s]$ and $[k]$, a partial results p_i can be computed as follows.

1. *Partial Randomize ElGamal*, using a random number r, $PartRandEG_r(\langle b, c\rangle) := \langle b^{t^{-1}} g^r, c^{t^{-1}} y^r\rangle$.

2. *Partial Re-Key ElGamal*, using $[k]$, pre-compute a secret sharing of $[k^{-1}]$. $\text{PartKeyEG}_{[k]}(\langle b, c\rangle) \coloneqq \langle b^{[k^{-1}]^{\pi}_i}, c^{t^{-1}}\rangle$.

3. *Partial Re-Shuffle ElGamal*, $\text{PartShuffleEG}_{[s]}(\langle b, c\rangle) \coloneqq \langle b^{[s]^{\pi}_i}, c^{[s]^{\pi}_i}\rangle$.

4. *Partial Re-KeyShuffle ElGamal*, using $[k]$ and $[s]$, pre-compute a secret sharing of $[sk^{-1}]$, $\text{PartKeyShuffleEG}_{[k],[s]}(\langle b, c\rangle) \coloneqq \langle b^{[sk^{-1}]^{\pi}_i}, c^{[s]^{\pi}_i}\rangle$.

Note that the pre-computations of new secret shares, such as computing $[sk^{-1}]$ from $[k]$ and $[s]$ only needs to be done once. Afterwards the secret shares of $[sk^{-1}]$ are stored, speeding up the computation in the future.

Besides distributing the trust put into a single TR over a group of TRs, the modified setting of Dist-PEP also removes the need for a trusted key server. Instead of letting a key server generate key-pair (x, y), this tasked can be performed by the group of TRs. Since x is just a random number, a secret sharing of random x with threshold t' can be generated. Picking $t' > t$ means that PEP operations can be performed by t TRs, while generating new private keys for participants requires a larger quorum.

5.3 POLYMORPHIC ACCESS MANAGEMENT

In this section we present our first contribution, that of Polymorphic Access Management (PACMAN). In PACMAN, data is encrypted using a modified version of ABE (Sec. 5.2.3.2), where the access tree is pseudonymised. The resulting construction has access structures, which look different at every party. This limits the amount of information these access structures leak, while preventing linkage of data structures across different parties. Pseudonymisation of access trees is performed by an intermediate party, the Transcryptor (TR). TR performs its operations on encrypted access trees, meaning it learns nothing about the attributes of the tree.

In Sec. 5.3.1, we first explain how this pseudonymisation process of access trees work. Afterwards, in Sec. 5.3.2 we explain our construction, which utilizes these access trees.

In the following section (Sec. 5.2.3.4), we show how PACMAN can be modified, distributing the role of TR.

5.3.1 *Pseudonymised Access Trees*

Let $\mathcal{T}$ be a tree representing an access structure. For every node x, we have a threshold value t_x and a set of children S_x. For every leaf node we have an associated attributed $att_x \in \mathbb{Z}_p$. Every attribute will look different at every party. This is achieved using pseudonymisation (Sec. 5.2.3.3).

We want to perform the pseudonymisation process on encryptions of attributes. This way the pseudonymisation can be performed by TR, without TR learning anything about identifiers and pseudonyms. For this we define the following operations for encrypting and decrypting trees. *Encrypt Tree* takes an access tree, and encrypts all its contents, creating an encrypted access tree. Let Y be all the leafs, and $T = \{\langle t_x, S_x\rangle : x \in \mathcal{T}\}$ be all the thresholds values and sets of children of all nodes in $\mathcal{T}$. $\text{EncTree}_y(\mathcal{T})$ first encrypts T as $W = \text{EncEG}_y(T)$. Afterwards, it takes every leaf y of $\mathcal{T}$, and computes $eatt_y = \text{EncEG}_y(att_y)$. The encrypted tree becomes $ET = \langle W, \forall y \in Y : eatt_y\rangle$.

Decrypt Tree decrypts an encrypted tree. $\text{DecTree}_X(\langle W, \forall y \in Y : eatt_y\rangle)$ decrypts W and $eatt_y$ for every $y \in Y$ and reconstructs the access tree.

Next, we define four operations on encrypted trees. Given an encrypted tree $ET = \text{EncTree}_y(\mathcal{T}) = \langle W, \forall y \in Y : eatt_y\rangle$, where $y = g^X$, the operations are as follows.

1. *Re-Randomize Tree*, randomizes the encryption of all elements. Pick random r_W, and r_y for every y in Y, $\text{RandTree}(\langle W, \forall y \in Y : eatt_y\rangle) \coloneqq \langle \text{RandEG}_{r_W}(W), \forall y \in Y : \text{RandEG}_{r_y}(eatt_y)\rangle$.

2. *Re-Key Tree*, re-encrypts the elements of the tree so that that can be decrypted using secret key $X = X \cdot k$. $\text{KeyTree}_k(\langle W, \forall y \in Y : eatt_y\rangle) \coloneqq \langle \text{KeyEG}_k(W), \forall y \in Y : \text{KeyEG}_k(eatt_y)\rangle$.

3. *Re-Shuffle Tree*, Re-Shuffles the attributes of the tree. $\text{ShuffleTree}_s(\langle W, \forall y \in Y : eatt_y\rangle) \coloneqq \langle W, \forall y \in Y : \text{ShuffleEG}(eatt_y)\rangle$. Afterwards $eatt_y$ will decrypt to $\text{DecEG}_X(att_y)^s$.

4. *Re-KeyShuffle Tree*, combines the Re-Shuffle and Re-Key operation. $\text{KeyShuffleTree}_{k,s}(\langle W, \forall y \in Y : eatt_y\rangle) \coloneqq \langle \text{KeyEG}_k(W), \forall y \in Y : \text{KeyShuffleEG}_{k,s}(eatt_y)\rangle$.

5.3.2 *Our Construction*

- *Setup* Let G_0 and G_1 be multiplicative groups of prime order p, with g being a generator of G_0 and bilinear map $e : G_0 \times G_0 \to G_1$. A trusted authority pick random α, β and X, computes master key MK and publishes public key PK.

$$\text{MK} = \langle \beta, g^\alpha, X\rangle$$

$$\text{PK} = \langle G_0, g, y = g^X, h = g^\beta, f = g^{\beta^{-1}}, e(g,g)^\alpha\rangle$$

- *Key Generation*, generates a key SK using set of attributes S, key-factor k and pseudonym-factor s. Pick $\ell \in_R \mathbb{Z}_p$, for every attribute $j \in S$ pick $r_j \in_R \mathbb{Z}_p$. The secret key becomes

$$\begin{aligned}\text{KeyGenPM}_{\text{MK}}(k, s, S) = \langle X' = X \cdot k, D = f^{(\alpha+\ell)k}, \\ \forall j \in S : D_j = g^{\ell} \cdot j^{r_j}, D'_j = g^{r_j} \rangle.\end{aligned} \tag{17}$$

  Note that by storing ℓ, additional attributes can be added to the same secret key at a later point in time. Given additional attribute i, compute $D_i = g^{\ell} \cdot i^{r_i}, D'_i = g^{r_i}, r_i \in_R \mathbb{Z}_p$.

- *Encrypt PACMAN*, encrypts M under the access structure defined in $\mathcal{T}$. Pick $w \in_R \mathbb{Z}_p$, construct polynomials using $\mathcal{T}$ as for an ordinary CP-ABE encryption (Sec. 5.2.3.2). Return ciphertext CT as

$$\begin{aligned}\text{EncPM}_{\text{PK}}(M, \mathcal{T}) = \langle ET = \text{EncTree}_y(\mathcal{T}), \\ \tilde{C} = M e(g, g)^{\alpha w}, C = h^w, \\ \forall y \in Y : C_y = g^{q_y(0)}, C'_y = att_y^{q_y(0)} \rangle.\end{aligned} \tag{18}$$

- *Decrypt PACMAN*, decrypts an encryption CT. Decrypt ET to obtain the pseudonymised access tree $\mathcal{T}'$. Let γ be the set of attributes in SK. If γ satisfies $\mathcal{T}'$, SK can be used to decrypt root node r as in the decryption process of CP-ABE (Sec. 5.2.3.2), $A = \text{DecryptNode}(CT, SK, r)$. Message M can be retrieved as $M' = \frac{\tilde{C}}{e(C,D)A^{-1}}$.

5.3.3 *Performing Pseudonymisation*

Given an ordinary ABE encryption E, performing the pseudonymisation operations on the access trees alone is not enough to provide security. The access structure only informs the receiver about which attributes to use for decryption. A party missing the decryption key for the access tree, but still holding the right attributes for satisfying the tree, can decrypt by trying the combination of all his decryption attributes on E. In our construction this is solved by using key-factor k in the key generation and decryption process. We define the following four operations on PACMAN encryptions.

1. *Re-Randomize PACMAN*, takes an encryption CT, and $r \in_R \mathbb{Z}_p$, and produces a randomized encryption of the same message.

$$\begin{aligned}\text{RandPM}_r(CT) \coloneqq \langle \underline{ET} = \text{RandTree}(ET), \\ \underline{\tilde{C}} = \tilde{C} \cdot e(g, g)^{\alpha r}, \\ \underline{C} = C \cdot h^r, \forall y \in Y : \\ \underline{C_y} = C_y \cdot g^r, \underline{C'_y} = C'_y \cdot att_y^r \rangle.\end{aligned} \tag{19}$$

2. *Re-Key PACMAN*, takes an encryption CT which can be decrypted using SK and produces an encryption which can be decrypted using SK_A.

$$\begin{aligned} \text{KeyPM}_r(CT) \coloneqq \langle \underline{ET} = \text{KeyTree}_k(ET), \\ \underline{C} = C^{k^{-1}}, \tilde{C}, \\ \forall y \in Y : C_y, C'_y \rangle. \end{aligned} \tag{20}$$

3. *Re-Shuffle PACMAN*, takes an encryption CT and re-shuffles the access tree so every the attribute of every leaf node y becomes $att'_y = att^s_y$.

$$\begin{aligned} \text{ShufflePM}_s(CT) \coloneqq \langle \underline{ET} = \text{ShuffleTree}_s(ET), C, \\ \tilde{C}, \forall y \in Y : C_y, C'_y \rangle. \end{aligned} \tag{21}$$

4. *Re-KeyShuffle PACMAN*, takes an encryption CT and creates an encryption which can be decrypted using key with key-factor k and pseudonym-factor s.

$$\begin{aligned} \text{KeySufflePM}_{k,s}(CT) \coloneqq \langle \underline{ET} = \text{KeyShuffleTree}_{k,s}(ET), \\ \underline{C} = C^{k^{-1}}, \\ \tilde{C}, \forall y \in Y : C_y, C'_y \rangle. \end{aligned} \tag{22}$$

5.3.4 *Storing and Retrieving Data*

In our construction the access structure is pseudonymised, meaning all the leaf-nodes will look different at every party. However, the internal nodes remain visible and identical. This allows us to use the access tree for storage indexing and performing binary search queries.

Let A be a participant wanting to store data at storage facility CP. A encrypts the data using access tree $\mathcal{T}$ and sends the encryption to TR. TRs perform the Re-KeyShuffle operation, and sends the result to CP. CP can decrypts in order to obtain a pseudonymised access tree. This tree is used as an index to store the data. Note that CP does not hold the attributes to satisfy the tree, thus he can not decrypt the data.

Let B be a participant with pseudonymised attributes γ. B wishes to retrieve all data from CP which can be satisfied using a subset γ' of γ. B encrypts the pseudonymised attributes in γ' and sends these to TR. TRs Re-KeyShuffle the encrypted attributes and send them to CP. CP can decrypt the attributes, resulting in pseudonymised attributes at CP. CP uses the pseudonymised attributes to retrieve any data for which the attributes satisfy the access trees. CP sends this data to TR, who Partial Re-Keys the data, such that B can decrypt. TR sends the results to B, who is able to use his key decrypt the data.

5.4 DISTRIBUTED POLYMORPHIC ATTRIBUTE-BASED ENCRYPTION

Similiar to how the role of TR in PEP can be distributed, the role of TR in PACMAN can also be distributed. We first give a description of how to perform Re-Randomization, Re-Keying, Re-Shuffling and Re-KeyfShuffling of encrypted access trees in a using a single TR. Afterwards, we modify our construction of PACMAN to allow perform the same operations in a distributed manner.

5.4.1 *Distributed Pseudonymised Access Trees*

We give a distributed version of the pseuodnymisation operations on encrypted access trees, allowing the pseudonymisation process to be performed in a distributed manner. For every participant j, a key-factor and pseudonym-factor are shared among the TRs using Shamir's Secret Sharing, as $[k_j]$ and $[s_j]$ respectively.

Given a quorum π of t TRs, each member i of π computes a partial result of the format $\mathrm{ET}_i = \langle W_i, \forall y \in Y : eatt_{y,i} \rangle$. The *Combine Tree* operation combines a set P of t partial results, $P = \{\mathrm{ET}_i : i \in \pi\}$. $\mathrm{CombTree}_\pi(P) \coloneqq \langle \mathrm{CombEG}_\pi(\{W_i : i \in \pi), \forall y \in Y : \mathrm{CombEG}_\pi(\{eatt_{y,i} : i \in \pi\}) \rangle$.

Given an encrypted tree $\mathrm{ET} = \langle W, \forall y \in Y : eatt_y \rangle$, the partial results ET_i for the different pseudonymisation operations can be computed as follows.

1. *Partial Re-Randomize Tree*, pick random r_W and r_y for every $y \in Y$,

$$\begin{aligned} \mathrm{PartRandTree}(\mathrm{ET}) \coloneqq \langle & \mathrm{PartRandEG}_{r_W}(W), \\ & \forall y \in Y : \mathrm{PartRandEG}_{r_y}(eatt_y) \rangle \end{aligned} \tag{23}$$

2. *Partial Re-Key Tree*, using secret sharing of key-factor,

$$\begin{aligned} \mathrm{PartKeyTree}_k(\mathrm{ET}) \coloneqq \langle & \mathrm{PartKeyEG}_k(W), \\ & \forall y \in Y : \mathrm{PartKeyEG}_k(eatt_y) \rangle. \end{aligned} \tag{24}$$

3. *Partial Re-Shuffle Tree*, using secret sharing of pseudonym-factor, let $W = \langle b, c \rangle$.

$$\begin{aligned} \mathrm{PartShuffleTree}_(\mathrm{ET}) \coloneqq \langle & \langle b^{-t}, c^{-t} \rangle, \\ & \forall y \in Y : \mathrm{PartShuffleEG}(eatt_y) \rangle \end{aligned} \tag{25}$$

4. *Partial Re-KeyShuffle Tree*, combines the re-key and re-shuffle operation,

$$\begin{aligned} \mathrm{PartKeyShuffleTree}_{[k],[s]}(\mathrm{ET}) \coloneqq \langle & \mathrm{PartKeyEG}_{[k]}(W), \\ & \forall y \in Y : \mathrm{PartKeyShuffleEG}_{[k],[s]}(eatt_y) \rangle \end{aligned} \tag{26}$$

5.4.2 *Our Construction*

We take the construction of ordinary PACMAN(Sec. 5.3) as a base and modify it so that the role of the TR becomes distributed. In addition, we remove the need for a trusted party for key generation. Instead keys are jointly created by the group of TRs, using Secret Sharing (Sec. 5.2.3.1). The master key becomes

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

From this the TRs collaborate and publish the public key

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

In order to perform the polymorphic operations in a distributed manner, we need to share key and pseudonym-factors among the TRs. For every participant i the TRs share a key-factor $[k_i]$ and a pseudonym-factor $[s_i]$.

Note that the threshold for secret sharings can differ. The secret sharings of the MK require a higher secured, and can thus be shared using a higher threshold than the key and pseudonym-factors.

5.4.3 *Performing Pseudonymisation*

Let π be a quorum of TRs of size t. We modify the pseudonymisation operations in ordinary PACMAN, so that they can be performed in a distributed manner by the TRs. For this each TR first computes a partial result p_i. We define two *Combine Polymorphic Attribute-Based Encryption* methods for combining these partial results, returning an encryption equal to performing the operation in a non-distributed manner. The first method combines partial results for the re-key, re-shuffle and re-keyshuffle operations. These partial results are of the form $p_i = \langle ET_i, \tilde{C}, C_i, \forall y \in Y : C_y, C'_y \rangle$. Let $P = \{p_i : i \in \pi\}$,

$$\begin{aligned} \text{CombPM1}_{\pi}(P) \coloneqq \langle ET = \text{CombTree}_{\pi}(P), \\ \tilde{C}, C = \prod_{j \in \pi} C_j, \forall y \in Y : C_y, C'_y \rangle. \end{aligned} \tag{27}$$

The second method combines partial results of the re-randomize operation, which have the form $p_i = \langle ET_i, \tilde{C}_j, C_i, \forall y \in Y : C_{y,i}, C'_{y,i} \rangle$, let $P = \{p_i : i \in \pi\}$,

$$\begin{aligned} \text{CombPM2}_{\pi}(P) \coloneqq \langle ET = \text{CombTree}_{\pi}(P), \tilde{C} = \prod_{j \in \pi} \tilde{C}_j, \\ C = \prod_{j \in \pi} C_j, \forall y \in Y : C_y = \prod_{j \in \pi} C_{y,j}, C'_y = \prod_{j \in \pi} C'_{y,j} \rangle. \end{aligned} \tag{28}$$

The partial results p_i can be computes using the following four methods. Let CT be a ciphertext encrypted under access tree $\mathcal{T}$, where Y are the leaves of $\mathcal{T}$.

1. *Partial Re-Randomize PACMAN*, pick random r and compute

$$\begin{aligned}\text{PartRandPM(CT)}_r \coloneqq \langle &\text{ET}_i = \text{PartRandTree}(\mathcal{T}),\\ &\tilde{C}_i = e(g,g)^{\alpha r} \cdot h^r, C_i = C \cdot \tilde{C},\\ &\forall y \in Y : C_{y,i} = C_y \cdot g^r, C'_{y,i} = C_y \cdot att_y^r\rangle.\end{aligned} \tag{29}$$

2. *Partial Re-Keys PACMAN*, uses secret sharing of key-factor $[k]$,

$$\begin{aligned}\text{PartKeyPM}_{[k]}(\text{CT}) \coloneqq \langle &\text{ET}_i = \text{PartKeyTree}_{[k]}(\mathcal{T}),\\ &\tilde{C}, C_i = C^{[k^{-1}]_i^\pi}, \forall y \in Y : C_y, C'_y\rangle.\end{aligned} \tag{30}$$

3. *Partial Re-Shuffles PACMAN*, uses secret sharing of pseudonym-factor $[s]$,

$$\begin{aligned}\text{PartShufflePM}_{[s]}(\text{CT}) \coloneqq \langle &\text{ET}_j i = \text{PartShuffleTree}_{[s]}(\mathcal{T}),\\ &\tilde{C}, C_i = C^{t^{-1}}, \forall y \in Y : C_y, C'_y\rangle.\end{aligned} \tag{31}$$

4. *Partial KeyShuffle PACMAN*, combines the re-key and re-shuffle operation,

$$\begin{aligned}&\text{PartKeyShufflePM}_{[k],[s]}(\text{CT}) \coloneqq\\ &\quad\langle \text{ET}_i = \text{PartKeyShuffleTree}_{[k],[s]}(\mathcal{T}),\\ &\quad\tilde{C}, C_i = C^{[k]_j^\pi}, \forall y \in Y : C_y, C'_y\rangle.\end{aligned} \tag{32}$$

5.4.4 *Computational Complexity*

In Table 4 we compare the computational complexity of the homomorphic operations in terms of multiplications, exponentiations, and bilinear map operations. Note that for PEP and Dist-PEP, we require two ElGamal encryptions, one for encrypting the identifier, a second for encrypting the data. When encrypting, multiplications are performed in the message space, which is computationally cheaper. We see that distributing both PEP and PACMAN requires little extra computational complexity. This is mainly due to parallelization of the distributed operations. For PEP and Dist-PEP all except the combine operation run in constant time. With the use of polymorphic access trees in PACMAN, these operations run linear in the amount of leaves of the access tree.

5.4.5 *Round Complexity*

Let A be a party who wants to send data D to B. This is done with the help of the Transcryptor(s), who will apply a homomorphic operation (Re-Randomize, Re-Key, Re-Shuffle or Re-KeyShuffle). All construction can be seen as a distributed (t, n) algorithm, where the non-distributed constructions have n and s set to one (1). Furthermore,

Table 4: Computational Complexity

Operation	PEP		Dist-PEP	
	Mult	Exp	Mult	Exp
Encryption	2*	4	2*	4
Decryption	2	2	2	2
Re-Randomize	2	2	2	4
Re-Key	0	1	0	2
Re-Shuffle	0	2	0	2
Re-KeyShuffle	1	2	0	2
Combine	0	0	2t	–

Operation	PACMAN			Dist-PACMAN		
	Mult	Exp	BilMap	Mult	Exp	BilMap
Encryption	$\ell + 2^*$	$5\ell + 4$	0	$\ell + 2^*$	$4\ell + 4$	0
Decryption	$\eta^2 + 2\ell + 3$	$\eta^2 + 2\ell + 4$	$2\ell + 1$	$\eta^2 + 2\ell + 3$	$\eta^2 + 2\ell + 4$	$2\ell + 1$
Re-Randomize	2ℓ	$4\ell + 2$	0	2ℓ	$4\ell + 2$	0
Re-Key	0	$\ell + 1$	0	0	$2\ell + 2$	0
Re-Shuffle	0	2ℓ	0	0	$2\ell + 2$	0
Re-KeyShuffle	ℓ	2ℓ	0	ℓ	$2\ell + 1$	0
Combine	–	–	–	$2t\ell + \ell$ $4t\ell + 2\ell$	0 0	0 0

*Computation performed on plaintext

note that partial results will be computed in parallel. All constructions require two rounds in order to perform a homomorphic operation as demonstrated in the following analysis. Round 1) A encrypts D as CT, picks a group π of t Transcryptors and sends CT to every member of π. Round 2) Every $j \in \pi$ performs the homomorphic operation and sends the (partial) result to B. B combines the partial results using CPA.

5.4.6 *Communication Complexity*

Table 5: Communication Complexity

Construction	Data transmission		Maximum bandwidth	
PEP	8ρ	$\mathcal{O}(\rho)$	4ρ	$\mathcal{O}(\rho)$
Dist-PEP	$8\rho t$	$\mathcal{O}(\rho t)$	$4\rho t$	$\mathcal{O}(\rho t)$
PACMAN	$8\rho\ell + 4\rho$	$\mathcal{O}(\rho\ell)$	$4\rho\ell + 2\rho$	$\mathcal{O}(\rho\ell)$
Dist-PACMAN	$8\rho\ell t + 4\rho t$	$\mathcal{O}(\rho\ell t)$	$4\rho\ell t + 2\rho t$	$\mathcal{O}(\rho\ell t)$

Let A hold an encryption D. A wants to perform a homomorphic operation (Re-Randomize, Re-Key, Re-Shuffle or Re-KeyShuffle) using secret values held by the Transcryptor(s). We analyze the total data transmission and maximum bandwith of all protocols. Table 5 shows the communication complexity in terms of total data transmission and the maximum bandwith (maximum amount of data in transit at a single time). Note that PEP requires two ElGamal encryptions, one for encrypting an identifier/pseudonym, and one for encrypting the actual data. Furthermore, we ignore the cost of transmitting the structure of an access tree. That is the t_x and S_x values for every node in an access tree. The size of this data highly depends on implementation, and is negligible.

5.4.7 *Storage Complexity*

Table 6: Ciphertext size

Construction	Encryption Size	
PEP	4ρ	$\mathcal{O}(\rho)$
Dist-PEP	4ρ	$\mathcal{O}(\rho)$
PACMAN	$4\rho\ell + 2\rho$	$\mathcal{O}(\rho\ell)$
Dist-PACMAN	$4\rho\ell + 2\rho$	$\mathcal{O}(\rho\ell)$

In this section we give a brief analysis of the storage complexity required for the different constructions.

Table 6 shows an overview of the size of encryption. We see that storage for the ordinary and distributed constructions is equal. However, for PACMAN the storage requirements is linear in the amount of leaves ℓ in the access tree.

Let n be the number of TRs, m the number of participants and u the total number of attributes. In the non-distributed setting $n = 1$. In PEP a key server stores the master key X. TR stores a pseudonym-factor s_i and key-factor k_i for every participant i. Furthermore, every participant i holds a decryption key X_i. This gives a storage complexity of $3m + 1 = \mathcal{O}(m)$ at the TR, and $\mathcal{O}(1)$ at every participant.

In Dist-PEP the pseudonym-factors and key-factors are distributed among the TRs. This means that for every participant i, every TR needs to hold a share of $[s_i]$ and $[k_i]$. Furthermore, for any 2 participants who want to exchange data some secret shares need to be precomputated. $[\sigma_{A\to B}] = [s_A^{-1} k_A s_B k_B^{-1}]$, $[\sigma_{B\to A}] = [s_B^{-1} k_B s_A k_A^{-1}]$, $[k_{A\to B}] = [k_A k_B^{-1}]$ and $[k_{B\to A}] = [k_B k_A^{-1}]$. This gives us a storage complexity of $\mathcal{O}(nm^2)$ at the TRs, and $\mathcal{O}(1)$ at each participant.

In PACMAN, for every participant i the TR needs to store a pseudonym-factor s_i and key-factor k_i. Furthermore every i needs to hold on to their own key, which can have a maximum size of $\mathcal{O}(u)$. This gives a total complexity of $\mathcal{O}(um)$ at the TR, and $\mathcal{O}(1)$ at each participant.

Dist-PACMAN requires the same sharing of pseudonym-factors and key-factors among the TRs as Dist-PEP. The storage requirements for keys are the same as in PACMAN. This gives us a total complexity of $\mathcal{O}(nm^2 + um)$ at the TRs, and $\mathcal{O}(u)$ at the participants.

Note that in an implementation, the given complexities are unlikely to be reached. In practice it is unlikely for ever participant to require data transmission with every participant. Also every participant will only hold a small subset of the attributes.

5.5 SECURITY ANALYSIS

5.5.1 *Scheme Security*

The security of the given schemes rely in the Discrete Logarithm (DL) prolem, the Computational Diffie-Hellman (CDH) problem and the Decisional Diffie-Helman (DDH) problem.

Let $\mathbb{G}$ be a group of prime order p with generator g.

- *DL* states that given given $g, a \in \mathbb{G}$, it is computationally infeasible to find x such that $g^x = a$.
- *CDH* states that given $g, g^a, g^b \in \mathbb{G}$ for any $a, b \in \mathbb{Z}_p$, it is hard to compute g^{ab}.
- *DDH* states that given given $g, g^a, g^b, g^c \in \mathbb{G}$ it is computationally hard to determine if $ab = c$.

All constructions rely on ElGamal encryption. Security of ElGamal relies on DL and CDH for its security and DDH for its semantic security. This means that within our chosen group all three of these problems should be hard. The construction for ABE requires bilinear maps. It is important to note that given two groups $\mathbb{G}_0$ and $\mathbb{G}_1$ with bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$, DDH is easy in $\mathbb{G}_0$. DDH can be decided by checking if $e(g^a, g^b) = e(g^c, g)$, which would imply whether $ab = c$. Therefore ElGamal should be performed either in $\mathbb{G}_1$, or in a different group where DDH is hard, in order to provide semantic security.

The distributed constructions achieve the distribution through Shamir Secret Sharing. Security of Shamir Secret Sharing relies on the property that given t points, infinite polynomials of degree t can be drawn through these points. This shows that given $t-1$ points, an adversary learns nothing about the polynomial, and by extension, learns nothing about the shared secret. Furthermore if DL is hard, the value $g^{[x]_j^\pi}$ leaks no information about $[x]_j^\pi$, nor x itself.

The security of attributes and their pseudonyms is the same as the security of identifiers and their pseudonyms. Let s_A and s_B be the pseudonym factors for A and B respectively. Let ID be an identifier with pseudonyms $\text{ID@A} = \text{ID}^{s_A}$ and $\text{ID@B} = \text{ID}^{s_B}$. Since DL is hard, it is hard to find any relation between the identifier and its pseudonyms.

5.5.2 *Privacy Analysis*

The construction should not leak any information about the data being handled. This means that though TRs facilitate in the exchange of data, they should not learn anything about its contents. A party storing an encryption for which it does not hold the decryption attributes, should learn as little as possible from this encryption.

By encrypting under access structures, the encrypting party has exact control over who is allowed to retrieve and decrypt the data. This allows the data owner to grant parties access to subsets of his data, while preventing these parties from collaborating and combining this data. The pseudonymisation process ensures that given two pseudonyms of the same attribute, these can not be linked. This would require the help of at least t TRs. This means that as long as the data itself doesn't reveal anything about the identity of the data owner, the data owner can not be linked. This allows the storage of encrypted data at a CP, without the need for a trusted party to provide access management. Through the use of pseudonymisation, these access structures will look different at every party. This means CP learns nothing about the meaning of an access structure, even when collaborating). CP is able to link data encrypted under the same attributes. However, this leaks no meaningful information, and gives us the abil-

ity to use the access structure as a means of querying the stored data, providing an efficient way of storing and retrieving the data.

It is possible to give a participant encryption attributes so he can only encrypt under a certain access structure. Though the participant can deviate from this access structure, he is able to remove attributes from it. However, this means data can only be encrypted under a more restrictive access structure. This means no data gets leaked, however this does remove visibility of the data for parties of which attributes get removed from the access structure.

Any group π of adversaries is able to pick value y and exploit TRs to obtain exponentiations of y by $k_i^{-1}, k_i k_j^{-1}, s_i$ and s_i^{-1}, for any $i, j \in \pi$. However, this leaks no information not already known by any individual member of π.

By distributing the pseudonymisation process, we remove the risk caused by a malicious TR. In the original PEP, a malicious TR collaborating with any other party would result in the potential leakage of all data. In our distributed setting, as long as at most $t - 1$ TRs become malicious, no data is leaked. If t TRs collaborate, they are able to reveal the non pseudonymized access structures, however, the same risk is present in Dist-PEP. Furthermore, they are not able to decrypt any extra data, something which would be possible in Dist-PEP.

Let t' be the threshold of the secret sharing of the master key MK. If t' TRs collaborate, they are able to generate decryption keys for all attributes, allowing them to retrieve all data. The amount of TRs n can easily be increased after setup, however, increasing t' is expensive. Because of this it is important to pick the right value for t' at setup.

5.6 FURTHER IMPROVEMENTS

5.6.1 *Enforcing Access Structures*

Let $\mathcal{T}$ be an access tree, with γ being the set of attributes at the leaves of $\mathcal{T}$. In order to encrypt a message under $\mathcal{T}$, the encrypting party needs to know the values of the attributes in γ. However, this means the encrypting party is able to use γ to construct a different access tree $\mathcal{T}'$, and use it for encryption.

This can be prevented as follows. Let A be a party holding attributes satisfying $\mathcal{T}$. A wants to delegate B to encrypt under $\mathcal{T}$. Let Y be the set of leaves of $\mathcal{T}$. A creates an encryption of message $M = 1$ as $K = \mathcal{EA}_{PK}(1, \mathcal{T}) = \langle ET, \tilde{C}, C, \forall y \in Y : C_y, C'_y \rangle$. A sends K to B through the TRs, who Re-KeyShuffle the encrypted tree ET. B receives $K' = \langle ET@B, \tilde{C}, C, \forall y \in Y : C_y, C'_y \rangle$. B can use K' to create an encryption of message M' under $\mathcal{T}$. B picks $w \in_R \mathbb{Z}_p$ and computes the ciphertext as $\langle ET@B, \tilde{C}^w \cdot M', C^w, \forall y \in Y : C_y^w, C_y'^w \rangle$. K' can be re-used to encrypt other messages. Though B is able to remove

leaves from the access tree, B is not able to modify the threshold or the inner structure of $\mathcal{T}$.

5.6.2 *Key Revocation and Time-Based Access Control*

In order to revoke a key, generated using key-factor k, the TRs simply stop using k in their Re-Key and Re-KeyShuffle operation. This means that once a party is no longer allowed to decrypt using a certain attribute, the TRs can simply create a key using a new key-factor, where the revoked attribute is removed. Afterwards, they stop using the old key-factor, rendering the old key with the revoked attribute useless.

This idea can be extended to add Time-Based Access Control to our construction. Let a be an attribute with expiration time T_a. Every participant i is given a decryption key with a in the set of attributes. When T_a is reached, the TRs stop using the existing key-factors for all participants. For every participant i the TRs generate a new random key-factor k_i'. A new key is generated using k_i, this time without attribute a and send to i. Encryptions will never be Re-Keyed to a key containing attributes for a.

5.7 PERFORMANCE ANALYSIS

Since the improvements suggested in Sec. 5.6 are specific to the distributed PACMAN setting, we provide a performance on the construction without these suggestions, in order to provide a fair comparison. We compare our proposed constructions (PACMAN and Dist-PACMAN) with the earlier constructions of PEP and Dist-PEP in terms of computational, round, communication and storage complexity.

5.8 APPLICATION FOR MEDICAL DATA STORAGE

Let P be a patient who is attending multiple HCPs. P has some wearable devices which are gathering and storing measurements at storage facility CP. The HCPs are also storing information about P at CP. P wants to be able to access all data gathered about him. Furthermore, he wants to allow the HCPs to access any data which is relevant for the treatment they provide.

5.8.1 *Data Categorization*

In order to achieve this, a correct access structure needs to be generated. For this we categorize the data, and give each HCP decryption attributes for categories which are relevant to the type of healthcare

they provide. Let C be the set of all categories. Let r_P be an attribute given to any HCP at which P is registered as a patient. Let u_P be a unique attribute, only held by P. Now data can be encrypted under an access structure where a party holding either u_P, or both r_P and a specific combination of attributes in C can access it. A HCP holding attributes to the correct categories, will not hold the attribute r_P, meaning he can not access the data. As long as all HCPs encrypt under data under the correct access structure, all parties can only access data which is relevant to them, while P is able to access all data gathered about him. When P stops using the services of a certain HCP, this HCP is given a new key which does not include the attribute r_P. Methods for revoking keys and enforcing access structures are useful for preventing HCP from using keys for old attributes, or encrypting under the wrong access structure (Sec. 5.6).

5.8.2 *Emergency Attributes*

Imagine the situation where P is involved in some accident in which P becomes incapacitated. P is brought to a HCP at which P is not registered (E does not hold r_P). In this situation it might be useful to override the ordinary access structure, so that E is able to data gathered about P. For this we can encrypt the data using a special emergency attribute e_P, which is specific for P. Data is encrypted such that a party holding e_P can always decrypt it. Decryption keys for e_P can only be acquired through a special process, after which the HCP requesting e_P should be held accountable for any misuse. More complicated emergency attributes could be constructed, where relatives are able to delegate access, or where multiple emergency attributes are used, each granting access to only a subset of the data gathered about P.

5.8.3 *Sharing with Research Institutions*

The vast amount of data aggregated in a PHR could be a great source for research institutions. However, for this the data needs to be anonymised, and the patient has to give consent. The pseudonymisation process provides unlinkability through the access structures, however, it is important that the data itself is also anonymous. If a patient P consents to data sharing with research institutions, data anonymisation can be performed by a HCP or the P. Afterwards the anonymised data gets encrypted such that the research institutions is granted access. If the research institutions during their research find any anomaly or health concern in P's data, the pseudonymisation process of the access structure can be reversed. If the access structure is of the form as described in Sec. 5.8.1, this should reveal r_P, an attribute unique to P, allowing the research institution to inform P.

5.9 CONCLUSION AND DISCUSSION

In this work we have constructed a method of encryption, where multiple parties are able to store data encrypted at a storage facility. Access to this data is managed through access structures, embedded in the encryption. This means that access management is simplified to the distribution of the correct attributed to each party. The data owner has fine-grained access managements, with efficient key revocation, and time-based access control. Furthermore, access structures are pseudonymised, meaning the access structure can not be used to link encryptions stored at different parties. This allows data storage at an untrusted cloud provider, while the pseudonymised access structure can be used to query this data. Any exchange of data is visible to the group of TRs, allowing logging and monitoring of all users. All of this is achieved without the need for any trusted or centralized party, such as a key server or transcryptor, or access manager.

Our distributed construction, is more robust than non-distributed systems. As long as at leat t TRs stay online the system remains operational. Also scalability is improved, when more bandwidth is required, it is always possible to introduce more TRs to the system.

Security of the construction relies on several long established crytographic principles, DL, CDH, DDH, and the security of Shamir secret sharing. And is shown to leak no data as long as at most $t-1$ TRs collaborate. Furthermore, given a threshold of t' for sharing the master key, as long as at most $t'-1$, the system only leaks the structure of access trees, but no data.

We have shown that the computational cost of the homomorphic operations on this form of encryption is linear dependant on the amount of leaves present in the access structure. Though the distributed system requires more storage of pre-computed values, the storage complexity is equal. Furthermore, this storage requirement is likely to be manageable in practice, since values only need to be precomputed for parties who require to exchange data, a number much smaller than that presented in our analysis.

Though the current construction using Shamir's Secret Sharing is effective, the threshold of the quorum is difficult to increase after setup. In a real-world implementation this might pose a problem. If in order to improve stability the total number of TRs needs to increase, so increases the risk of reaching a threshold of malicious TRs. A construction where the treshold of the quorum can be increased after setup would solve this. For this one should look into other methods multi-party computation, in order to achieve re-encryption and pseudonymisation.

Another problem which remains is that of enforcing the encryption under the correct access structure. Though we have shown that there are ways to enforce the use of a certain access structure, the encrypt-

ing party can still choose to use a more restricting version of this access structure. However, if the access structure is chosen correctly, this behavior will either be visible, or not relevant to the data owner. Still, in order to further prevent such behavior, one approach is to instead use KP-ABE[79]. This would reduce the cost of encryption and encryption, and allow the change of access policy after encryption. However this again moves access management towards the TRs, and comes with new complexity and security challenges of its own.

The current work uses a re-keying process where there has to be a relationship between the original key and the new key of the ciphertext. There are re-encryption schemes where no relationship between the keys are required[71]. The trade-of is that in this scheme the secret key of the original encryption has to be shared among a group of intermediate parties (in our context the group of TRs). However, this approach has the potential to reduce the amount storage required in our current construction, such as the storage of key-factors. Furthermore, this approach could provide even more flexibility in key-revocation and key management. To further explore this idea, a concrete design for such a scheme has to be made, allowing a comparison to be made in terms of its performance, security and efficiency.

6

CONCLUSION AND DISCUSSION

The adaption of SMDs, and the emergence of e-health over the recent years, show promise for a new healthcare future. In this future, SMDs share their data with HCPs, allowing the HCP to be aided by automated systems to monitor the patient's health. This can reduce the HCP's workload, medical costs, and improve the quality of healthcare itself. Personally, I am convinced this future will arrive, one way or another, however, how much remains of an individuals privacy is uncertain.

In this work we have shown that both in the Netherlands and Europe, lawmaking is aimed at creating a secure and privacy-preserving landscape for e-health. Unfortunately this lawmaking is slow, resulting in e-health which fails to achieve the benefits it promises to bring, and as a result chooses privacy compromises in its implementation.

Besides legislation, the field of cryptography is also struggling with e-health. Though promising methods for securing the handling of medical data have been proposed, so far the field has not been able to solve all the challenges which come with e-health and a PHR. Each method comes with a costs, such as computational overhead in SMC, increased complexity, or reliance on a trusted party in others.

However, possibly the biggest party to blame is the industry of SMDs. Manufacturers and cloud providers choose to exploit the data SMDs gather, in ways which would not be acceptable for medical data. Because of this many chose to label their data as non-medical. Even manufacturers who market their devices for healthcare purposes, show little effort to truly protect the data they handle as such.

If no action is taken, e-health will grow into a system where the promises it brings, are instead used as an excuse to further reduce the little privacy that remains in the fast growing digital world. With this prospect, this thesis aims to push e-health towards a more optimistic direction, by attempting to answer the following research question,

How can data gathered by SMDs be securely shared with HCPs, without loss of privacy?

With the contributions made in this work, we have moved closer to realizing a secure PHR. We propose PACMAN, a method for managing a distributed PHR, where multiple parties, such as HCPs, SMDs, and patient, are able to contribute data to a single PHR. To the best of our knowledge, this is the first e-health system fulfilling all of the following properties. The patient has fine-grained control over which parties have access to which parts of his data. All data in the patient's

PHR, and all actions on this data, are visible to the patient. Furthermore, the system allows temporary access rights, and full revocation of access. All of this is achieved without the need for a central authority, while remaining resistant to collusion by a group of parties. Through pseudonymisation colluding parties are unable to link the data owner, while a threshold of colluding parties is needed to break any of the privacy and security guarantees.

PACMAN is suitable for the e-health architecture currently present in the Netherlands, and could be implemented without the need for introducing or removing any parties. Currently data exchange is managed by the LSP, which itself is split into different segments. The role of the transcryptors can be assigned to these segments. The cloud providers can continue to provide the same service, however they will longer be able to view their customer s data. Any analytical or computational service they provide, can be implemented using SMC, while other cryptographc methods allow for correct search and retrieval of any data. Furthermore, we have shown that complexity in terms of communication, storage and computation is manageable for this system.

With this we have shown that through cryptographic methods it is possible to create a secure environment, in which patient, HCP, cloud provider, and even research institutions, can benefit from the future that is e-health, without privacy concessions.

6.1 FUTURE WORK

Our contribution is promising, and demonstrates that the implementation of a PHR, where data gathered by SMDs can be shared with HCPs is possible, even without a central authority. However, as with all proposals for secure e-health, there are still some challenges.

The current system relies on a group of transcryptors, of which a quorum has to collaborate in order to assist in the exchange of data. To improve the bandwidth of the system, it is likely that in the future more transcryptors will be added. Unfortunately, increasing the size of the quorum is expensive and likely to be impractical. As the size of the quorum remains constant, the chance of a quorum collaborating for malicious purposes increases.

In our proposal it is possible to enforce that a HCP storing data in a patient's PHR can do so only under a certain access structure. However, the HCP is able to deviate and encrypt under a more restricting access structure. Fortunately, this action does not leak any information, however, possibly does hide it for certain parties. Furthermore, this action should be visible to the patient, after which the HCP can be held accountable.

Another limitation, is that modifying the access structure of data already stored at a cloud provider is expensive. Furthermore, there is

no method of ensuring that the cloud provider removes the original encryption under the old access structure.

Our solution assumes that all parties operate in a semi-trusted manner, meaning they will follow the protocol, but will try to learn as much other secret information as possible. Through methods such as proof of work, PACMAN can be extended to function even with non-trusted parties. This would come with extra computational overhead. However, by using proof of work selectively, critical applications can be made more secure.

Before PACMAN can be implemented, further analysis is required, in order to determine how it will perform in a real-world setting. Storage complexity is highly dependant on the amount of parties which need to exchange data. Because of this, both the current and future connectivity of parties needs to be investigated in order to give a strong estimation of it's storage requirements. Both the efficiency and the security of PACMAN is highly dependant on the threshold parameter: the required size of a quorum or transcryptors. Because of this, research has to put into finding parameters for PACMAN, suitable for the Dutch e-health landscape. Setting the threshold too high would slow down the system, while setting it too low would compromise security.

Besides improvements on the security of this work, there is room for extending it. Logging is currently performed by the group of transcryptors. However, this means that this log is distributed. Forming consensus and managing this distributed log creates new challenges. Solutions can be sought in consensus and distributed algorithm, but might also be found in the recent developments in blockchain.

Though our approach provides some anonymous search based on the structure of anonymised access trees, retrieving at which party data is stored still remains unsolved.

Currently the system is mainly focused on secure storage and access management, however achieves little in terms of SMC. If a cloud provider wishes to perform some analysis or diagnostics on data it stores, the current system only supports a naive approach. Here a party with access to the data would have to retrieve it from the cloud provider, after which the two can engage in a SMC protocol. Extending the system such that the cloud provider can perform secure computations, without the the need for a trusted party with access, would be a huge improvement to this approach. One approach would be to use encrypt the data using homomorphic encryption, after which the cloud provider can use it for its computations. The decryption key for the data would then be stored using PACMAN.

A

IEEE WIFS 2017 SUBMISSION

The contents of this appendix is a submission for IEEE WIFS currently under review.

A.1 ABSTRACT

The availability of wearable devices such as smart watches and fitness trackers are a recent development. Among other things these devices are able to measure the activity and vital signs of their wearers. As the types of data these devices are able to gather increases, the potential for them to be used as a source of data grows. This calls for a secure method of controlling the digital exchange of medical data between wearables and healthcare providers, and healthcare providers in general. By enforcing the exchange of data to go through a central authority, a patient can be given more control over who is able to access their medical data. This central authority is then given the task of monitoring access and ensuring that any access requirements are met. Though effective, this solution relies on a highly trusted central authority. In this work we propose a scheme using Polymorphic Encryption and Pseudonomysation and Shamir Secret Sharing in order to provide anonymous data storage and data exchange. Our proposal removes the need for a central authority, and instead uses a group of authorities, of which a quorum is needed to facilitate the exchange of data.

A.2 INTRODUCTION

In recent years there has been a rise in the amount of wearable devices, such as fitness trackers and smartwatches[3]. These devices which often work in conjunction with a smartphone are becoming more accurate in their measurements and will increasingly use cloud storage and become part of the internet of things[99]. The increased adaptation of these wearables goes hand in hand with Electronic Health Care (e-healthcare): The shift of health care to the digital domain. In an e-healthcare setting, patients are able to measure their medical condition at home, using sensors placed in wearable devices. The result is that medical measurements can be taken over longer period of time in an automated fashion. In this manner costs on the amount of hospital visits and work performed by a doctor can be reduced Furthermore the additional information gathered by these devices can improve the quality of a diagnosis.

Collecting and sharing medical data in e-healthcare would benefit greatly from a cloud-based solution, improving the availability of a patient's medical data and the ease of sharing. However, handling medical data in such a manner causes concern for privacy. Medical data is highly sensitive and a cloud Storage Provider (SP) should not be trusted with it. A natural solution would be to store this data in encrypted form, however, providing proper access control and anonymity become two important challenges. What is needed is a system where self monitoring devices and healthcare providers (HCPs) are able to share medical data, while providing proper access management and security.

Several countries such as the Netherlands allow a patient to decide who is allowed to (digitally) access their medical data. This is usually done through a Central Authority (CA). The patient tells the CA which HCPs are allowed to exchange his/her medical data, fully trusting the CA to not misuse this information, and furthermore monitor and assist in this exchange accordingly. Usually only certified HCPs are allowed to connect to this system. This means third parties, such as wearable vendors acting as SPs are unable to connect. The result is that if provided at all, every vendor provides a different method of sharing collected data with HCPs, making it complicated to use data collected by wearables in the medical domain.

Through methods such as end-to-end encryption[72], a party is able to provide storage without learning anything about the information stored. Using secure multi-party computations, such as garbled circuits[44, 45] and private linear branching programs[57][60], computations over the encrypted data can be performed. This allows for automated diagnosis, without revealing anything about the patient.

In Polymorphic Encryption and Pseudonymisation for Personalised Healthcare (PEP) by Verheul et al.[23] a patient and HCPs are able to store data at SPs in encrypted form through a central authority called the Transcryptor (TR). The SP learns nothing about the contents and owner of the data, while TR only learns who is sharing data. Here anonymity is provided through pseudonymisation, the act of using different identifiers for the same data at different parties. The TR is able to provide a mapping between these pesudonyms without learning anything about these pseudonyms. This way data exchange can only take place through TR, creating a central authority which is able to perform the required access control and access monitoring.

The PEP solution heavily relies on the TR as a crucial party. If TR goes offline the whole system stops working. Furthermore with devices automatically storing medical data, the scalability of such a centralized system becomes a concern. Finally if TR collaborates with a single SP or HCP, they can generate keys for decrypting any stored data.

In this work we propose a solution where the role of TR is distributed over a group of n TRs. A quorum of t TRs have to collaborate in order to provide the mapping of pseudonyms. This way any amount of TRs less than t is unable to learn decryption keys or any information about pseudonym mapping, even with the help of additional HCPs or SPs. Additionally this setup increases the scalability and availability of such a system, as long as t TRs are online the system will remain fully functional. The result is a more robust solution, more adapted to the real-world requirements of e-healthcare.

The rest of this paper is organized as follows. We introduce our cryptograhpic building blocks and the construction of PEP in Section A.3. In Section A.4 we describe our proposal, extending PEP to a distributed scheme. Section A.5 provides a complexity and run-time analysis of our proposal. Finally we conclude in Section A.6.

A.3 PRELIMINAIRIES

In this section we shall describe the cryptographic building blocks which form the basis of our proposed scheme. First a description of ElGamal is given. Next, we show how Shamir Secret Sharing can be used to share a secret among several parties, and show how computations can be performed using these secrets. Finally we describe Polymorphic Encryption and Pseudonymisation.

A.3.1 *Cryptographic building blocks*

A.3.1.1 *ElGamal Cryptosystem*

ElGamal is an asymetric cryptosystem relying on the difficulty of compute discrete logarithms over finite fields[49]. Let p and q be large primes such that $p = 2q + 1$. Let $g \in \mathbb{F}_p^*$ be a generator of multiplicative subgroup $\mathbb{G}_q$ of order q. The secret key is a random value $x < q$, from which the corresponding public key $y = g^x \mod p$ is derived. Encryption is performed as follows: $E_y(M) = \langle b, c\rangle = \langle g^r, y^r \cdot M\rangle \mod p$, where r is a random value smaller than q. To decrypt we compute $D_x(\langle b, c\rangle) = \frac{c}{b^x} = \frac{y^r \cdot M}{g^{rx}} = M \mod p$.

A.3.1.2 *Shamir Secret Sharing*

A (t, n) threshold scheme allows a secret S to be shared among n parties. S can not be recovered unless $t \leqslant n$ parties work together[50]. Let q be a prime number and $S < q$. Let f be a polynomial of degree $t - 1$ such that $f(0) = S$. Value S is shared among n parties by giving each party i a unique point (x_i, y_i) on f. x_i is public while y_i is kept secret.

Given t unique points on f, any point on the polynomial f can be computed using Lagrange interpolation. Let π be a sequence of t unique numbers in $\{1, ..., n\}$. In order to obtain point x we calculate

$$f(x) = \sum_{i \in \pi} \ell_i(x) \mod q, \tag{33}$$

$$\text{where } \ell_i(x) = y_i \prod_{j \in \pi, j \neq s} \frac{x - x_j}{x_i - x_j} \mod q. \tag{34}$$

Since $S = f(0)$, in order to retrieve secret value S we need to compute $f(0)$. Every party $i \in \pi$ calculates a partial result of the Lagrange interpolation as $S_i = \ell_i(0)$, which can then be combined at a single party using Eq. 33, resulting in S.

A.3.1.3 *Secret Sharing Operations*

Several operation on shared secrets can be performed, resulting in either a new secret sharing, or calculated value. Given two secret sharings, a secret sharing of the addition[98] or multiplication[51] of these secrets can be generated. Furthermore secret sharings of random values can be generated, in addition to a secret sharing of the inverse of a secret sharing[52]. Exponentiation by a shared secret can be performed without revealing the secret value[55].

- *Addition and multiplication of secrets.* Let S and T be two secrets which are shared among n parties using polynomials f and g. Let degree of f be $t_f - 1$ and degree of g be $t_g - 1$. $f(0) = S$ and $g(0) = T$. Every party i has a unique public value x_i and holds the corresponding secret values $f(x_i)$ and $g(x_i)$.

  In order to generate a secret sharing over polynomial $a = f + g$, with $a(0) = f(0) + g(0) = S + T$, every party i computes the value of $a(x_i) = f(x_i) + g(x_i)$ as their secret share of a. Resulting threshold of a is $t_a = \max(t_f, t_g)$.

  In order to generate a secret sharing of polynomial $h = f \cdot g$ with threshold t_h, a quorum of size $t_f + t_g - 1$ is required. Let π be a group of unique parties, with $|\pi| = t_f + t_g - 1$. Every party i computes $h'(x_i) = f(x_i) \cdot g(x_i)$. The degree of h' equals $t_f + t_g - 2$. In order to obtain a secret sharing with threshold t_h, every party i creates a secret sharing for $H'_i = h'(x_i) \prod_{j \in \pi, j \neq i} x_j$, giving every other party j share $H'_{i,j}$. After exchanging shares every i computes their secret share of h as $\sum_{j \in \pi} H'_{j,i}$.

- *Generate sharing of a secret random value.* In order to generate a (t, n) secret sharing of R, an unknown secret random value, every party i picks a random polynomial r_i of degree $t - 1$ and gives every party j a share $r_{i,j}$. Every j computes $\sum_{k=1}^{n} r_{i,j}$ as their secret share of R.

- *Inverse of a secret* S mod q. Let S be a (t, n) secret shared using polynomial f with $f(0) = S$. In order to compute the inverse of S mod q, t parties generate Z, a (t, n) sharing of a secret random value. Secret shares for $W = Z \cdot S$ are computed. The value of W is revealed and a secret sharing for $W^{-1}1$ is generated. Finally a secret sharing for $S^{-1} = W^{-1} \cdot Z$ can be calculated.

- *Exponentiation by a secret* S on a public value k, without revealing S can be performed as follows. Given π, a sequence of t unique numbers in $\{1, ..., n\}$, every member i in π calculates $S_i = \ell_i(0)$ using Eq. 34, followed by

$$R_i = k^{S_i}, \tag{35}$$

obtaining partial result R_i. These partial results are combined using

$$\begin{aligned} \prod_{s \in B} R_{\pi(s)} &= k^{S_{\pi(1)}} \cdot ... \cdot k^{S_{\pi(t)}} \\ &= k^{S_{\pi(1)} + ... + S_{\pi(t)}} = k^S, \end{aligned} \tag{36}$$

resulting in the exponentiation of k by S. Given k^S and k it is infeasible to obtain S, due to the Discrete Logarithm Problem.

A.3.2 *Polymorhic encryption and Pseudonymization*

Security in PEP comes from the fact that an individual is known by different identifiers at different parties. Anyone generating or accessing stored data is called a participant, which in the context of e-health are patients and HCPs. A fully trusted key server generates a private key x and publishes the corresponding public key $y = g^x$. Every participant A providing or accessing data about patient P holds its own related key pair (x_A, y_A) with secret key $x_A = x \cdot K_A$ and public key $y_A = y^{K_A}$. K_A is the key-factor for A, a fixed random value, only known by the TR.

Patient P is given a unique identifier $pid_P \in \mathbb{G}_q$. At every participant A, P will be known by a pseudonym of pid_P, which is a different related identifier $pid_P@A = pid_P^{S_A}$ (pid_P to the power S_A), S_A is the pseudonym-factor of A, a fixed random value, known only by the TR.

Taking advantage of the homomorphic properties of ElGamal, the TR is able to transformations between encrypted pseudonyms. Let p and q be large primes such that $p = 2q + 1$. Let $g \in \mathbb{F}_p^*$ be a generator of multiplicative subgroup $\mathbb{G}_q$ of order q.

A.3.2.1 *PEP Operations*

Let $epid_P = E_y(pid_P) = \langle b, c \rangle$, an encryption of the identifier of patient P. The following operations are defined:

- *re-randomisation* takes an encryption $\langle b, c\rangle$ and a random value $r < q$ and produces a randomized encryption of the same message,

$$RR_r(\langle b, c\rangle) \coloneqq \langle g^r \cdot b, y^r \cdot c\rangle \mod p. \tag{37}$$

- *re-keying* takes an encryption $\langle b, c\rangle$ encrypted using $pk = y = g^x$ and a value $k < q$ and produces an encryption which can be decrypted using different private key $x' = x \cdot k$,

$$RK_k(\langle b, c\rangle) \coloneqq \langle b^{k^{-1}}, c\rangle \mod p, \tag{38}$$

  where k^{-1} is the inverse of $k \mod q$.

- *re-shuffling* takes an encryption $\langle b, c\rangle$ of message m and a value $s < q$ and produces an encryption of m^s,

$$RS_s(\langle b, c\rangle) \coloneqq \langle b^s, c^s\rangle \mod p. \tag{39}$$

A.3.2.2 *Storing data*

An encrypted identifier $epid_P$ is stored at all participants supplying data for P. When a data supplier wants to store data D at storage facility SF, it first re-randomizes the $epid_P$ to form a polymorphic pseudonym, $PP = epid_P = RR_r(ppid_P)$ using random r. Next, it encrypts the data D to obtain $C = E_y(D)$. PP and C are sent to the TR who re-keys and re-shuffles PP using K_{SF} and S_{SF}, which results in the encrypted polymorphic pseudonym,

$$epid_P@SF = RS_{S_{SF}}(RK_{K_{SF}}(PP)). \tag{40}$$

TR sends $epid_P@SF$ and C to the SF. SF decrypts $epid_P@SF$ using his private key $D_{y_{SF}}(epid_P@SF) = pid_P@SF = pid_P^{S_{SF}}$. SF stores C using $pid_P@SF$ as a reference.

A.3.2.3 *Retrieving data*

Let A be a particpant authorized to retrieve data D stored encrypted as $C = E_y(D)$ at storage provider SF. Since A is eligible to retrieve D he knows $ppid_P = E_y(pid_P)$. A performs $PP = RR_r(ppid_P)$ using random r and sends PP to the TR. TR performs $epid_P@SF = RS_{S_{SF}}(RK_{K_{SF}}(PP))$ and sends the result to SF. SF decrypts $epid_P@SF$ to obtain $pid_P@SF$ and use that to look up the requested encrypted data C. SF re-randomizes C, $C' = RR_s(C)$ using random s and sends C' to TR. TR re-keys and re-shuffles C' using the key-factor and pseudonym-factor of A, $R = RK_{K_A}(C')$. R is send to A who can decrypt it, $D' = D_{x_A}(C') = D$.

Table 7: Symbols and their meaning.

Symbol	Description
x, y	master key pair
x_A, y_A	key pair of participant A
K_A	key-factor for participant A
S_A	pseudonym-factor for participant A
pid_A	personal identifier of patient P
$pid_P@A$	pseudonym of patient P at participant A
$epid_P@A$	encrypted Pseudonym of patient P at participant A

A.3.2.4 *Limitation of PEP*

Through the pseudonymization process, participants have to communicate through the TR. This creates a central party, who can perform access management, logging and monitoring. However this centralization and other design decisions create several security threats.

- The key server holds the master private key x. If key server collaborates with the storage facility, all data and all polymorphic pseudonyms can be decrypted.
- TR holds secrets K_i and S_i. Participant A holds $x_A = x \cdot K_A$. If TR collaborates with any participant A, they are able to learn x, creating the same security threat as compromising the key server. Furthermore they are able to compute any other private key, as $x_B = x \cdot K_B$. Furthermore they are able to learn personal identifier pid_j from any pseudonym of j.

To prevent malicious behaviour from key server and TR from, PEP proposes the use of Hardware Security Modules (HSMs). These HSMs manage cryptographic keys, key-factors and pseudonym-factors and all PEP operations can only be performed inside these HSMs. Though this prevents direct leakage of these secret values, TR is still able to transform any data or pseudonym, such that it can be decrypted by any participant.

Besides security threats, the construction of PEP can only be scaled vertically. Keeping the context of wearables in mind, we can only expect the amount of data which the TR needs to process to increase. Furthermore whenever TR goes offline, no exchange of data can take place.

A.4 DISTRIBUTED POLYMORPHIC ENCRYPTION AND PSEUDONYMISATION

In the original PEP a single TR uses several PEP operations described in Section A.3.2.1 in order to perform the required pseudonym mapping, and re-keying of encrypted data. In order to obtain our solution where these operations are instead performed by a quorum of t out of n TRs, we first show how these operations can be performed in a (t, n) manner (Section A.4.2). Next we demonstrate how new parties and TRs can be added to the scheme after setup (Section A.4.3 and A.4.4). Finally we suggest a way through which t' TRs can jointly perform key generation and key management, which would remove the need for a trusted key server (Section A.4.5).

A.4.1 *Setup*

For every participant i a key-factor K_i and pseudonym-factor S_i are secretly shared among n TRs, together with K_i^{-1} the inverse of K_i mod q. Let $K_{i,j}$, $S_{i,j}$ and K_i^{-1} be the partial results of the Lagrange interpolation of secret shares K_i, S_i and K_i^{-1} for TR j (Eq. 1).

A.4.2 (t, n) *PEP Operations*

Let $\text{epid}_i = E_y(\text{pid}_i) = \langle b, c \rangle$, an encryption of the identifier of patient i. Let π, a set of unique TRs in n with $|\pi| = t$. We describe how to compute partial results for the PEP operations described in Section A.3.2.1. These partial results can be combined using the *Combine Partial Results* operation. Given B, a set of partial results we compute

$$\text{CPR}(B) := \langle \prod_{j \in \pi} b_j, \prod_{j \in \pi} c_j \rangle. \tag{41}$$

- *partial re-randomisation*, every TR $j \in \pi$ picks a random value $r_j < q$ and computes partial result as

  $$\text{PRR}_{r_j}(\langle b, c \rangle) := \langle g^{r_j} \cdot b^{t^{-1}}, y^{r_j} \cdot c^{t^{-1}} \rangle, \tag{42}$$

  with t^{-1} being computed mod q. Combining these partial results gives a re-randomisation, equal to the operation $\text{RR}_{r'}(\langle b, c \rangle)$, with $r' = \sum_{j \in \pi} r_j$.

- *partial re-keying*, every TR $j \in \pi$ computes a partial result as

  $$\text{PRK}_{K_{i,j}}(\langle b, c \rangle) := \langle b^{K_{i,j}^{-1}}, c^{t^{-1}} \rangle, \tag{43}$$

  with t^{-1} being computed mod q. Combining these partial result gives a re-keying, equal to the operation $\text{RK}_{K_i}(\langle b, c \rangle)$.

- *partial re-shuffling*, every TR $j \in \pi$ computes a partial result as

$$\mathrm{PRS}_{S_{i,j}}(\langle b, c\rangle) \coloneqq \langle b^{S_{i,j}}, c^{S_{i,j}}\rangle. \tag{44}$$

  Combining these partial results gives a re-shuffling, equal to the operation $\mathrm{RS}_{S_i}(\langle b, c\rangle)$.

- *partial re-keyshuffling*, the operation of re-keying, followed by re-shuffling can be combined into a single step by pre-computing secret shares for the value $Q_i = K_i^{-1} \cdot S_i$, with $Q_{i,j}$ being partial result of the Lagrange interpolation of secret share Q_i for TR j (Eq. 1). Every TR $j \in \pi$ computes a partial result as

$$\mathrm{PRKS}_{Q_{i,j},S_{i,j}}(\langle b, c\rangle) \coloneqq \langle b^{Q_{i,j}}, c^{S_{i,j}}\rangle. \tag{45}$$

  Combining these partial results gives a re-keyshuffling, equal to the operation $\mathrm{RS}_{S_i}(\mathrm{RK}_{K_i}(\langle b, c\rangle))$.

A.4.3 *Generating new key and pseudonym-factors*

Let i be a new participant given permission to access some data owned by patient P. At least $2t-1$ TRs create two shared secrets of secret random values for key-factor K_i and pseudonym-factor S_i. They jointly compute secret sharings for K_i^{-1} and $Q_i = K^{-1} \cdot S_i$. For any TR not involved in the generation process, shares can be generated using Lagrange interpolation.

A.4.4 *Adding a new transcryptor*

After setup, it is possible to add or replace a TR. Let TR_k be a new TR joining the setup. Given a (t, n) secret sharing of secret $Z \in \mathbb{G}_q$, new shares for Z be generated using t TRs. Let π be a quorum of t TRs. First TR_k is assigned a unique point x_k not used by any other TR in n. Every TR $i \in \pi$ computes a partial result for the secret share of Z for k as

$$Z_{k,i} = Z_i \prod_{j \in \pi, j \neq i} \frac{x_k - x_j}{x_i - x_j} \mod q, \tag{46}$$

and sends $Z_{k,i}$ to k. k can compute his secret share of Z using

$$Z_k = \sum_{i \in \pi} Z_{k,i} \mod q. \tag{47}$$

This way k can be given secret shares of all pseudonym-factors, key-factors and secret shares of any pre-computations, such as the inverse of key-factors.

A.4.5 *Key generation*

The role of a trusted key server can instead be distributed among t' TRs. Secret key x becomes a (t', n) secret sharing of a secret random value. It is possible to perform exponentiation by a secret value, and thus the public key $y = g^x$ can be jointly computed and revealed.

For any participant i with (t, n) secretly shared key-factor K_i, $t' + t - 1$ TRs can collaborate to compute the private key x_i of i. First they compute shares of the multiplication of K_i and x. Next they can compute partial results for $x_i = K_i \cdot x$ and send them to i, who can compute his secret key x_i.

Note that throughout this process no TR ever learns nor holds the value of x. Furthermore the secret sharing of x can be done using t' greater than t, improving security of x. Also instead of giving all TRs secret shares for x, shares can be given only to TRs with increased trust.

A.5 ANALYSIS

In this section we shall present a security sketch of our scheme. This is followed by an analysis of its complexity. Finally using an implementation we test its performance.

A.5.1 *Security*

Shamir Secret Sharing guarantees that for a (t, n) secret sharing, nothing can be learned about the shared secret with less than t shares. This means with $t - 1$ collaborating TRs can not learn anything about any K_j or S_j. $t' - 1$ TRs are unable to determine anything about x. The unlinkability of randomized encryptions and randomized encrypted pseudonyms relies on the Decisional Diffie-Hellman problem: Given $g, g^a, g^b, g^c \in G$, it should be infeasible to determine if $a \cdot b = c$.

A.5.2 *Complexity*

We perform complexity analysis of our scheme based on the amount of multiplications and exponentiation performed. The CPR operation is analyzed for one participant, while the PRR, PRK, PRS and PRKS operations are analyzed as if they are performed by t TRs (See Table 8). All operations show a complexity of $O(t)$.

All PEP operations in the original scheme have complexity of $O(1)$. Note that though we provide a PRR operation, this operation is not used by the TRs in our scheme.

For any (t, n) PEP operation the message complexity analysis is as follows. A participant needs to send a polymorphic pseudonym and encrypted data C to t TRs. After performing their operation t

Table 8: Computational complexity (t, n) PEP operations.

Operation	**Multiplication**	**Exponentiation**	**Overall Complexity**
CPR	$2t$	0	$O(t)$
PRR	$2t$	$4t$	$O(t)$
PRK	0	$2t$	$O(t)$
PRS	0	$2t$	$O(t)$
PRKS	0	$2t$	$O(t)$

Table 9: Computation time of 1000 PRKS and CPR operations

t	PRKS computation time (ms)	CPR computation time (ms)
1	671	0
3	667	8
7	698	23
10	707	26
13	731	36
16	734	55
19	754	68
22	749	63
25	762	67
28	781	87
31	776	95
34	779	111
37	794	121
40	803	128

TRs send their partial result to a single receiver. This gives us a tight bound of $\theta(2t)$ messages.

A.5.3 *Performance*

The protocols have been implemented using C++ and the GMP library[1]. We test the performance on a machine running Windows 10.0, with a Intel Core i5-7200 running at 2.50GHHz. We analyze the computation time for performing t PRKS operations, and combining the partial results using CPR, for different values of t (See Table 9). The PKRS computation time is the average time spend by a single TR.

We plot the results in Fig. 7 and 8. As expected the both methods demonstrate a linear complexity of $O(t)$.

1 See https://gmplib.org/.

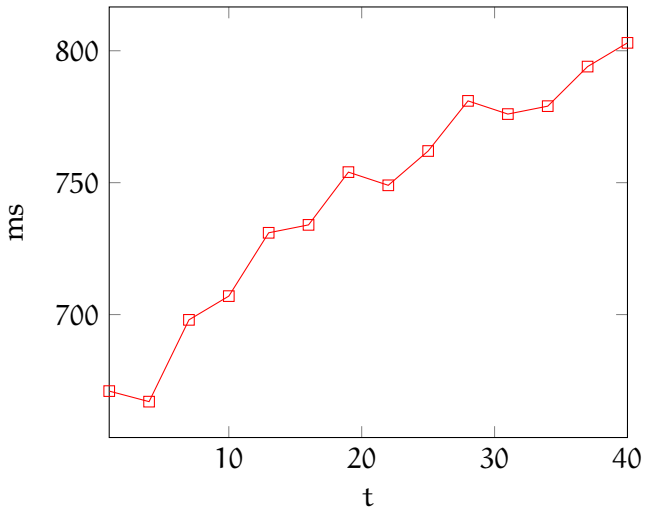

Figure 7: PKRS computation time

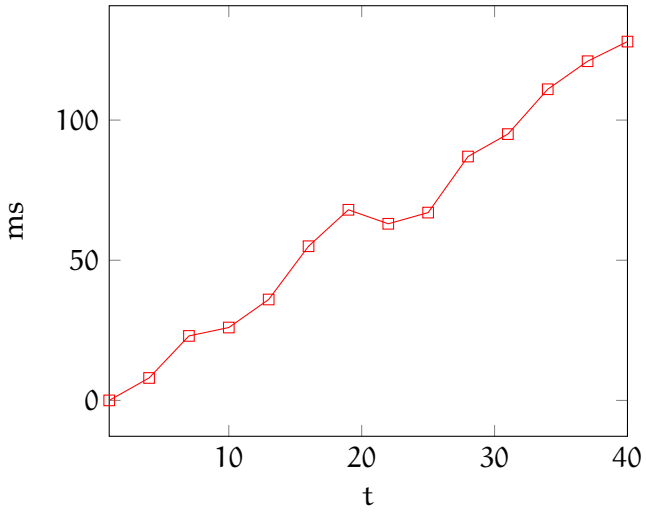

Figure 8: CPR computation time

A.6 CONCLUSION AND DISCUSSION

In this work we propose a solution for controlling access in a distributed manner. Through pseudonymisation, we simultaneously provide anonymity, while enforcing that any access to data is controlled and monitored. Furthermore, this is done in a distributed manner using Shamir Secret Sharing, increasing scalability and security. We further improve on previous solutions, by removing the need for a key server. We show that our solution performs well through complexity analysis and by measuring the performance of an actual implementation.

The current scheme has no way of detecting if any TR is not deviating from the protocol. Furthermore, since pseudonyms by nature are supposed to look random, there is no way of knowing if an obtained pseudonym is correct. To solve this, our scheme can be extended with non-interactive proofs[71] in order to demonstrate correctness of work performed by the TRs.

BIBLIOGRAPHY

[1] Forbes. *Worldwide Smartwatch Market Will See Modest Growth in 2016 Before Swelling to 50 Million Units in 2020, According to IDC*. http://www.idc.com/getdoc.jsp?containerId=prUS41736916.

[2] Forbes. *Gartner Says Worldwide Wearable Devices Sales to Grow 18.4 Percent in 2016*. http://www.gartner.com/newsroom/id/3198018.

[3] CC S Insight. *Wearables Market to be Worth $25 Billion by 2019*. http://www.ccsinsight.com/press/company-news/2332-wearables-market-to-be-worth-25-billion-by-2019-reveals-ccs-insight.

[4] Forbes. *Wearable Tech Market To Be Worth $34 Billion By 2020*. https://www.forbes.com/sites/paullamkin/2016/02/17/wearable-tech-market-to-be-worth-34-billion-by-2020/#309c3e493cb5.

[5] 2016 Q2 Tractica. *Wearable Devices for Healthcare Markets*. https://www.tractica.com/research/wearable-devices-for-healthcare-markets/. Q2, 2016 Q 2.

[6] *Owlet Smart Sock*. http://www.owletcare.com/.

[7] Mirza Mansoor Baig, Hamid Gholamhosseini, and Martin J. Connolly. "A comprehensive survey of wearable and wireless ECG monitoring systems for older adults." In: *Med. Biol. Engineering and Computing* 51.5 (2013), pp. 485–495. DOI: 10.1007/s11517-012-1021-6. URL: https://doi.org/10.1007/s11517-012-1021-6.

[8] S. Seneviratne, Y. Hu, T. Nguyen, G. Lan, S. Khalifa, K. Thilakarathna, M. Hassan, and A. Seneviratne. "A Survey of Wearable Devices and Challenges." In: *IEEE Communications Surveys Tutorials* PP.99 (2017), pp. 1–1. DOI: 10.1109/COMST.2017.2731979.

[9] Shyamal Patel, Hyung Park, Paolo Bonato, Leighton Chan, and Mary Rodgers. "A review of wearable sensors and systems with application in rehabilitation." In: *Journal of NeuroEngineering and Rehabilitation* 9.1 (2012), p. 21. ISSN: 1743-0003. DOI: 10.1186/1743-0003-9-21. URL: https://doi.org/10.1186/1743-0003-9-21.

[10] *Kardia, Heart health at your fingertips*. statline.cbs.nl/Statweb/publication/?VW=T&DM=SLNL&PA=83037NED&D1=a&D2=&HD=160517-1200&HDR=G1&STB=T.

[11] Forbes. *Behind Epic Systems' Alliance with Apple*. https://www.forbes.com/sites/zinamoukheiber/2014/06/04/behind-epic-systems-alliance-with-apple/#52ddf3e06b2f.

[12] Luigi Atzori, Antonio Iera, and Giacomo Morabito. "The Internet of Things: A survey." In: *Computer Networks* 54.15 (2010), pp. 2787–2805. DOI: 10.1016/j.comnet.2010.05.010. URL: https://doi.org/10.1016/j.comnet.2010.05.010.

[13] A. Lymberis. "Smart wearables for remote health monitoring, from prevention to rehabilitation: current R D, future challenges." In: *4th International IEEE EMBS Special Topic Conference on Information Technology Applications in Biomedicine, 2003*. 2003, pp. 272–275. DOI: 10.1109/ITAB.2003.1222530.

[14] P. Yanchapaxi, C. Tipantuña, and X. Calderón. "Wearable system for monitoring of human physical activities." In: *2017 Fourth International Conference on eDemocracy eGovernment (ICEDEG)*. 2017, pp. 245–250. DOI: 10.1109/ICEDEG.2017.7962543.

[15] Hoang T. Dinh, Chonho Lee, Dusit Niyato, and Ping Wang. "A survey of mobile cloud computing: architecture, applications, and approaches." In: *Wireless Communications and Mobile Computing* 13.18 (2013), pp. 1587–1611. ISSN: 1530-8677. DOI: 10.1002/wcm.1203. URL: http://dx.doi.org/10.1002/wcm.1203.

[16] *Vereniging Zorgaanbieding Voor Zorgcommunicatie (VZVZ)*.

[17] *Apple Privacy Policy*. https://www.apple.com/privacy/privacy-policy/.

[18] *Google Privacy Policy*. https://static.googleusercontent.com/media/www.google.com/nl//intl/nl/policies/privacy/google_privacy_policy_nl.pdf.

[19] *Fitbit Privacy Policy*. https://www.fitbit.com/legal/privacy-policy.

[20] *Philips Privacy Policy*. https://www.usa.philips.com/a-w/privacy-notice.html.

[21] *Withings Privacy Policy*. http://media-cdn.withings.com/termsandconditions/EU/2015-Privacy-Policy-VEE.pdf?__t=20170226T163500%200100.

[22] *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.

[23] Eric R. Verheul, Bart Jacobs, Carlo Meijer, Mireille Hildebrandt, and Joeri de Ruiter. "Polymorphic Encryption and Pseudonymisation for Personalised Healthcare." In: *IACR Cryptology ePrint Archive* 2016 (2016), p. 411. URL: http://eprint.iacr.org/2016/411.

[24] VZVZ. *10 feiten over het LSP*. https://www.vzvz.nl/page/Zorgconsument/Links/Over-VZVZ/10-feiten-over-het-LSP.

[25] "Zorgvisie ict magazine, nr. 3." In: ().

[26] *Folder, Uw medische gegevens beschikbaar via het Landelijk Schakelpunt (LSP)*. https://www.vzvz.nl/uploaded/FILES/htmlcontent/Voorlichtingsmateriaal/Folder%20Nederlands.pdf.

[27] U.S. Government Publishing Office. *H. REPT. 104-736 - HEALTH INSURANCE PORTABILITY AND ACCOUNTABILITY ACT OF 1996*. https://www.gpo.gov/fdsys/search/pagedetails.action?granuleId=CRPT-104hrpt736&packageId=CRPT-104hrpt736. 1996.

[28] Medical Economics. *Patient records: The struggle for ownership*. http://medicaleconomics.modernmedicine.com/medical-economics/news/patient-records-struggle-ownership.

[29] *Wet algemene bepalingen burgerservicenummer*. Wet- en regelgeving, http://wetten.overheid.nl/BWBR0022428/2014-01-06.

[30] "Wet gebruik burgerservicenummer in de zorg (30.380)." In: *publicatie wet (Staatsblad 2008, nr. 164)* (2008).

[31] "Cliëntenrechten bij elektronische verwerking van gegevens." In: *publicatie wet (Staatsblad 2016, nr. 373)* (2017).

[32] "Cliëntenrechten bij elektronische verwerking van gegevens." In: *publicatie wet (Staatsblad 2017, nr. 279)* (2016).

[33] "XVI Volkgsgezondheid, Welzijn en Sport Rijksbegroting 2016." In: (2016).

[34] *KNMG-richtlijn, omgaan met medische gegevens*. 2016.

[35] *https://developers.google.com/fit/terms*. https://developers.google.com/fit/terms.

[36] Jason Andress. *The basics of information security - understanding the fundamentals of InfoSec in theory and practice, Second edition*. Syngress, 2014. ISBN: 0128008121. URL: http://www.worldcat.org/oclc/880706587.

[37] Mike Ryan. "Bluetooth: With Low Energy Comes Low Security." In: *7th USENIX Workshop on Offensive Technologies, WOOT '13, Washington, D.C., USA, August 13, 2013*. 2013. URL: https://www.usenix.org/conference/woot13/workshop-program/presentation/ryan.

[38] Kassem Fawaz, Kyu-Han Kim, and Kang G. Shin. "Protecting Privacy of BLE Device Users." In: *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*. 2016, pp. 1205–1221. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/fawaz.

[39] Martin M. Merener. "Theoretical Results on De-Anonymization via Linkage Attacks." In: *Trans. Data Privacy* 5.2 (2012), pp. 377–402. URL: http://www.tdp.cat/issues11/abs.a074a11.php.

[40] "Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data." In: *Official Journal of the European Communities, L 281, 23 November 1995* (1995).

[41] Assad Abbas and Samee Ullah Khan. "A Review on the State-of-the-Art Privacy-Preserving Approaches in the e-Health Clouds." In: *IEEE J. Biomedical and Health Informatics* 18.4 (2014), pp. 1431–1441. DOI: 10.1109/JBHI.2014.2300846. URL: https://doi.org/10.1109/JBHI.2014.2300846.

[42] Michael O. Rabin. "How To Exchange Secrets with Oblivious Transfer." In: *IACR Cryptology ePrint Archive* 2005 (2005), p. 187. URL: http://eprint.iacr.org/2005/187.

[43] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems." In: *Commun. ACM* 21.2 (1978), pp. 120–126. DOI: 10.1145/359340.359342. URL: http://doi.acm.org/10.1145/359340.359342.

[44] Andrew Chi-Chih Yao. "How to Generate and Exchange Secrets (Extended Abstract)." In: *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*. 1986, pp. 162–167. DOI: 10.1109/SFCS.1986.25. URL: http://dx.doi.org/10.1109/SFCS.1986.25.

[45] Riccardo Lazzeretti and Mauro Barni. "Private Computing with Garbled Circuits [Applications Corner]." In: *IEEE Signal Process. Mag.* 30.2 (2013), pp. 123–127. DOI: 10.1109/MSP.2012.2230540. URL: http://dx.doi.org/10.1109/MSP.2012.2230540.

[46] Pascal Paillier. "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes." In: *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*. 1999, pp. 223–238. DOI: 10.1007/3-540-48910-X_16. URL: https://doi.org/10.1007/3-540-48910-X_16.

[47] Ivan Damgård and Mads Jurik. "A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System." In: *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Cheju Island, Korea, February 13-15, 2001, Proceedings*. 2001, pp. 119–136. DOI: 10.1007/3-540-44586-2_9. URL: https://doi.org/10.1007/3-540-44586-2_9.

[48] Shafi Goldwasser and Silvio Micali. "Probabilistic Encryption." In: *J. Comput. Syst. Sci.* 28.2 (1984), pp. 270–299. DOI: 10.1016/0022-0000(84)90070-9. URL: https://doi.org/10.1016/0022-0000(84)90070-9.

[49] Taher El Gamal. "A public key cryptosystem and a signature scheme based on discrete logarithms." In: *IEEE Trans. Information Theory* 31.4 (1985), pp. 469–472. DOI: 10.1109/TIT.1985.1057074. URL: https://doi.org/10.1109/TIT.1985.1057074.

[50] Adi Shamir. "How to Share a Secret." In: *Commun. ACM* 22.11 (1979), pp. 612–613. DOI: 10.1145/359168.359176. URL: http://doi.acm.org/10.1145/359168.359176.

[51] J. Willemson D. Bogdanov and S. Laur. "How to Securely Perform Computation on Secret-Shared Data." In: (2007).

[52] Judit Bar-Ilan and Donald Beaver. "Non-Cryptographic Fault-Tolerant Computing in Constant Number of Rounds of Interaction." In: *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing, Edmonton, Alberta, Canada, August 14-16, 1989*. 1989, pp. 201–209. DOI: 10.1145/72981.72995. URL: http://doi.acm.org/10.1145/72981.72995.

[53] Ivan Damgård and Rune Thorbek. "Linear Integer Secret Sharing and Distributed Exponentiation." In: *IACR Cryptology ePrint Archive* 2006 (2006), p. 44. URL: http://eprint.iacr.org/2006/044.

[54] Matt Blaze, Gerrit Bleumer, and Martin Strauss. "Divertible Protocols and Atomic Proxy Cryptography." In: *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*. 1998, pp. 127–144. DOI: 10.1007/BFb0054122. URL: https://doi.org/10.1007/BFb0054122.

[55] Yvo Desmedt and Yair Frankel. "Threshold Cryptosystems." In: *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*. 1989, pp. 307–315. DOI: 10.1007/0-387-34805-0_28. URL: https://doi.org/10.1007/0-387-34805-0_28.

[56] Mauro Barni, Pierluigi Failla, Vladimir Kolesnikov, Riccardo Lazzeretti, Ahmad-Reza Sadeghi, and Thomas Schneider. "Secure Evaluation of Private Linear Branching Programs with Medical Applications." In: *Computer Security - ESORICS 2009, 14th European Symposium on Research in Computer Security, Saint-Malo, France, September 21-23, 2009. Proceedings*. 2009, pp. 424–439. DOI: 10.1007/978-3-642-04444-1_26. URL: https://doi.org/10.1007/978-3-642-04444-1_26.

[57] Mauro Barni, Pierluigi Failla, Riccardo Lazzeretti, Annika Paus, Ahmad-Reza Sadeghi, Thomas Schneider, and Vladimir Kolesnikov. "Efficient privacy-preserving classification of ECG signals." In: *First IEEE International Workshop on Information Forensics and Security, WIFS 2009, London, UK, December 6-9, 2009*. 2009, pp. 91–95.

[58] Ahmad-Reza Sadeghi and Thomas Schneider. "Generalized Universal Circuits for Secure Evaluation of Private Functions with Application to Data Classification." In: *Information Security and Cryptology - ICISC 2008, 11th International Conference, Seoul, Korea, December 3-5, 2008, Revised Selected Papers*. 2008, pp. 336–353. DOI: 10.1007/978-3-642-00730-9_21. URL: https://doi.org/10.1007/978-3-642-00730-9_21.

[59] Mauro Barni, Claudio Orlandi, and Alessandro Piva. "A privacy-preserving protocol for neural-network-based computation." In: *Proceedings of the 8th workshop on Multimedia & Security, MM&Sec 2006, Geneva, Switzerland, September 26-27, 2006*. 2006, pp. 146–151. DOI: 10.1145/1161366.1161393. URL: http://doi.acm.org/10.1145/1161366.1161393.

[60] Mauro Barni, Pierluigi Failla, Riccardo Lazzeretti, Ahmad-Reza Sadeghi, and Thomas Schneider. "Privacy-Preserving ECG Classification With Branching Programs and Neural Networks." In: *IEEE Trans. Information Forensics and Security* 6.2 (2011), pp. 452–468. DOI: 10.1109/TIFS.2011.2108650. URL: https://doi.org/10.1109/TIFS.2011.2108650.

[61] Justin Brickell, Donald E. Porter, Vitaly Shmatikov, and Emmett Witchel. "Privacy-preserving remote diagnostics." In: *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*. 2007, pp. 498–507. DOI: 10.1145/1315245.1315307. URL: http://doi.acm.org/10.1145/1315245.1315307.

[62] Z. R. Li, E. C. Chang, K. H. Huang, and F. Lai. "A secure electronic medical record sharing mechanism in the cloud computing platform." In: *2011 IEEE 15th International Symposium on Consumer Electronics (ISCE)*. 2011, pp. 98–103. DOI: 10.1109/ISCE.2011.5973792.

[63] Yu-Yi Chen, Jun-Chao Lu, and Jinn-ke Jan. "A Secure EHR System Based on Hybrid Clouds." In: *J. Medical Systems* 36.5 (2012), pp. 3375–3384. DOI: 10.1007/s10916-012-9830-6. URL: https://doi.org/10.1007/s10916-012-9830-6.

[64] Mohammad Jafari, Reihaneh Safavi-Naini, and Nicholas Paul Sheppard. "A rights management approach to protection of privacy in a cloud of electronic health records." In: *Proceedings of the 11th ACM Workshop on Digital Rights Management, Chicago, Illinois, USA, October 21, 2011*. 2011, pp. 23–30. DOI: 10.1145/2046631.2046637. URL: http://doi.acm.org/10.1145/2046631.2046637.

[65] John Pecarina, Shi Pu, and Jyh-Charn Liu. "SAPPHIRE: Anonymity for enhanced control and private collaboration in healthcare clouds." In: *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, CloudCom 2012, Taipei, Taiwan, December 3-6, 2012*. 2012, pp. 99–106. DOI: 10.1109/CloudCom.2012.6427488. URL: https://doi.org/10.1109/CloudCom.2012.6427488.

[66] John Linwood Griffin, Trent Jaeger, Ronald Perez, Reiner Sailer, Leendert van Doorn, and Ramón Cáceres. "Trusted Virtual Domains: Toward Secure Distributed Services." In: *Proceedings of the First Conference on Hot Topics in System Dependability*. HotDep'05. Yokohama, Japan: USENIX Association, 2005, pp. 4–4. URL: http://dl.acm.org/citation.cfm?id=1973400.1973404.

[67] Hans Löhr, Ahmad-Reza Sadeghi, and Marcel Winandy. "Securing the e-health cloud." In: *ACM International Health Informatics Symposium, IHI 2010, Arlington, VA, USA, November 11 - 12, 2010, Proceedings*. 2010, pp. 220–229. DOI: 10.1145/1882992.1883024. URL: http://doi.acm.org/10.1145/1882992.1883024.

[68] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. "Improved proxy re-encryption schemes with applications to secure distributed storage." In: *ACM Trans. Inf. Syst. Secur.* 9.1 (2006), pp. 1–30. DOI: 10.1145/1127345.1127346. URL: http://doi.acm.org/10.1145/1127345.1127346.

[69] Varad Kirtane and C. Pandu Rangan. "RSA-TBOS signcryption with proxy re-encryption." In: *Proceedings of the 8th ACM Workshop on Digital Rights Management, Alexandria, VA, USA, October 27, 2008*. 2008, pp. 59–66. DOI: 10.1145/1456520.1456531. URL: http://doi.acm.org/10.1145/1456520.1456531.

[70] Elena Kirshanova. "Proxy Re-encryption from Lattices." In: *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*. 2014, pp. 77–94. DOI: 10.1007/978-3-642-54631-0_5. URL: https://doi.org/10.1007/978-3-642-54631-0_5.

[71] Markus Jakobsson. "On Quorum Controlled Asymmetric Proxy Re-encryption." In: *Public Key Cryptography, Second International Workshop on Practice and Theory in Public Key Cryptography, PKC '99, Kamakura, Japan, March 1-3, 1999, Proceedings*. 1999, pp. 112–121. DOI: 10.1007/3-540-49162-7_9. URL: https://doi.org/10.1007/3-540-49162-7_9.

[72] Kurt Rohloff and Yuriy Polyakov. "An end-to-end security architecture to collect, process and share wearable medical device data." In: *17th International Conference on E-health Networking, Application & Services, HealthCom 2015, Boston, MA, USA, October 14-17, 2015*. 2015, pp. 615–620. DOI: 10.1109/HealthCom.2015.7454578. URL: https://doi.org/10.1109/HealthCom.2015.7454578.

[73] Eu-Jin Goh, Hovav Shacham, Nagendra Modadugu, and Dan Boneh. "SiRiUS: Securing Remote Untrusted Storage." In: *Proceedings of the Network and Distributed System Security Symposium, NDSS 2003, San Diego, California, USA*. 2003. URL: http://www.isoc.org/isoc/conferences/ndss/03/proceedings/papers/9.pdf.

[74] Mahesh Kallahalla, Erik Riedel, Ram Swaminathan, Qian Wang, and Kevin Fu. "Plutus: Scalable Secure File Sharing on Untrusted Storage." In: *Proceedings of the FAST '03 Conference on File and Storage Technologies, March 31 - April 2, 2003, Cathedral Hill Hotel, San Francisco, California, USA*. 2003. URL: http://www.usenix.org/events/fast03/tech/kallahalla.html.

[75] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. "Improved proxy re-encryption schemes with applications to secure distributed storage." In: *ACM Trans. Inf. Syst. Secur.* 9.1 (2006), pp. 1–30. DOI: 10.1145/1127345.1127346. URL: http://doi.acm.org/10.1145/1127345.1127346.

[76] Sabrina De Capitani di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. "Over-encryption: Management of Access Control Evolution on Outsourced Data." In: *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007*. 2007, pp. 123–134. URL: http://www.vldb.org/conf/2007/papers/research/p123-decapitani.pdf.

[77] Amit Sahai and Brent Waters. "Fuzzy Identity Based Encryption." In: *IACR Cryptology ePrint Archive* 2004 (2004), p. 86. URL: http://eprint.iacr.org/2004/086.

[78] Mehdi Sookhak, F. Richard Yu, Muhammad Khurram Khan, Yang Xiang, and Rajkumar Buyya. "Attribute-based data access control in mobile cloud computing: Taxonomy and open issues." In: *Future Generation Comp. Syst.* 72 (2017), pp. 273–287.

DOI: 10.1016/j.future.2016.08.018. URL: https://doi.org/10.1016/j.future.2016.08.018.

[79] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. "Attribute-based encryption for fine-grained access control of encrypted data." In: *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*. 2006, pp. 89–98. DOI: 10.1145/1180405.1180418. URL: http://doi.acm.org/10.1145/1180405.1180418.

[80] John Bethencourt, Amit Sahai, and Brent Waters. "Ciphertext-Policy Attribute-Based Encryption." In: *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*. 2007, pp. 321–334. DOI: 10.1109/SP.2007.11. URL: https://doi.org/10.1109/SP.2007.11.

[81] Ming Li, Shucheng Yu, Yao Zheng, Kui Ren, and Wenjing Lou. "Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption." In: *IEEE Trans. Parallel Distrib. Syst.* 24.1 (2013), pp. 131–143. DOI: 10.1109/TPDS.2012.97. URL: https://doi.org/10.1109/TPDS.2012.97.

[82] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. "Attribute based data sharing with attribute revocation." In: *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2010, Beijing, China, April 13-16, 2010*. 2010, pp. 261–270. DOI: 10.1145/1755688.1755720. URL: http://doi.acm.org/10.1145/1755688.1755720.

[83] Kan Yang, Xiaohua Jia, and Kui Ren. "Attribute-based fine-grained access control with efficient revocation in cloud storage systems." In: *8th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '13, Hangzhou, China - May 08 - 10, 2013*. 2013, pp. 523–528. DOI: 10.1145/2484313.2484383. URL: http://doi.acm.org/10.1145/2484313.2484383.

[84] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. "Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing." In: *INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15-19 March 2010, San Diego, CA, USA*. 2010, pp. 534–542. DOI: 10.1109/INFCOM.2010.5462174. URL: https://doi.org/10.1109/INFCOM.2010.5462174.

[85] Junbeom Hur and Dong Kun Noh. "Attribute-Based Access Control with Efficient Revocation in Data Outsourcing Systems." In: *IEEE Trans. Parallel Distrib. Syst.* 22.7 (2011), pp. 1214–1221. DOI: 10.1109/TPDS.2010.203. URL: https://doi.org/10.1109/TPDS.2010.203.

[86] Melissa Chase and Sherman S. M. Chow. "Improving privacy and security in multi-authority attribute-based encryption." In: *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*. 2009, pp. 121–130. DOI: 10.1145/1653662.1653678. URL: http://doi.acm.org/10.1145/1653662.1653678.

[87] Huang Lin, Zhenfu Cao, Xiaohui Liang, and Jun Shao. "Secure Threshold Multi Authority Attribute Based Encryption without a Central Authority." In: *Progress in Cryptology - INDOCRYPT 2008, 9th International Conference on Cryptology in India, Kharagpur, India, December 14-17, 2008. Proceedings*. 2008, pp. 426–436. DOI: 10.1007/978-3-540-89754-5_33. URL: https://doi.org/10.1007/978-3-540-89754-5_33.

[88] Thomas Hupperich, Hans Löhr, Ahmad-Reza Sadeghi, and Marcel Winandy. "Flexible patient-controlled security for electronic health records." In: *ACM International Health Informatics Symposium, IHI '12, Miami, FL, USA, January 28-30, 2012*. 2012, pp. 727–732. DOI: 10.1145/2110363.2110448. URL: http://doi.acm.org/10.1145/2110363.2110448.

[89] Sushmita Ruj, Milos Stojmenovic, and Amiya Nayak. "Privacy Preserving Access Control with Authentication for Securing Data in Clouds." In: *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012, Ottawa, Canada, May 13-16, 2012*. 2012, pp. 556–563. DOI: 10.1109/CCGrid.2012.92. URL: https://doi.org/10.1109/CCGrid.2012.92.

[90] Ming Li, Shucheng Yu, Kui Ren, and Wenjing Lou. "Securing Personal Health Records in Cloud Computing: Patient-Centric and Fine-Grained Data Access Control in Multi-owner Settings." In: *Security and Privacy in Communication Networks - 6th International ICST Conference, SecureComm 2010, Singapore, September 7-9, 2010. Proceedings*. 2010, pp. 89–106. DOI: 10.1007/978-3-642-16161-2_6. URL: https://doi.org/10.1007/978-3-642-16161-2_6.

[91] Ming Li, Shucheng Yu, Yao Zheng, Kui Ren, and Wenjing Lou. "Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption." In: *IEEE Trans. Parallel Distrib. Syst.* 24.1 (2013), pp. 131–143. DOI: 10.1109/TPDS.2012.97. URL: https://doi.org/10.1109/TPDS.2012.97.

[92] Jin Li, Qian Wang, Cong Wang, Ning Cao, Kui Ren, and Wenjing Lou. "Fuzzy Keyword Search over Encrypted Data in Cloud Computing." In: *INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15-19 March 2010, San Diego, CA, USA*. 2010, pp. 441–445. DOI: 10.1109/INFCOM.2010.

5462196. URL: http://dx.doi.org/10.1109/INFCOM.2010.5462196.

[93] Cong Wang, Kui Ren, Shucheng Yu, and Karthik Mahendra Raje Urs. "Achieving usable and privacy-assured similarity search over outsourced cloud data." In: *Proceedings of the IEEE INFOCOM 2012, Orlando, FL, USA, March 25-30, 2012*. 2012, pp. 451–459. DOI: 10.1109/INFCOM.2012.6195784. URL: http://dx.doi.org/10.1109/INFCOM.2012.6195784.

[94] Juan Ramón Troncoso-Pastoriza, Stefan Katzenbeisser, and Mehmet Utku Celik. "Privacy preserving error resilient dna searching through oblivious automata." In: *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*. 2007, pp. 519–528. DOI: 10.1145/1315245.1315309. URL: http://doi.acm.org/10.1145/1315245.1315309.

[95] M. Li, S. Yu, N. Cao, and W. Lou. "Authorized Private Keyword Search over Encrypted Data in Cloud Computing." In: *2011 31st International Conference on Distributed Computing Systems*. 2011, pp. 383–392. DOI: 10.1109/ICDCS.2011.55.

[96] Y.T. Zhang K. Hung and B. Tai. "Wearable medical devices for tele-home healthcare." In: *Engineering in Medicine and Biology Society* (2004), pp. 5384 –5387. DOI: 10.1109/IEMBS.2004.1404503.

[97] Ming Li, Shucheng Yu, Yao Zheng, Kui Ren, and Wenjing Lou. "Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption." In: *IEEE Trans. Parallel Distrib. Syst.* 24.1 (2013), pp. 131–143. DOI: 10.1109/TPDS.2012.97. URL: https://doi.org/10.1109/TPDS.2012.97.

[98] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract)." In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*. 1988, pp. 1–10. DOI: 10.1145/62212.62213. URL: http://doi.acm.org/10.1145/62212.62213.

[99] Tom Page. "A Forecast of the Adoption of Wearable Technology." In: *IJTD* 6.2 (2015), pp. 12–29. DOI: 10.4018/IJTD.2015040102. URL: https://doi.org/10.4018/IJTD.2015040102.