Bachelor Thesis

An Attribute-based Encryption System in the context of a Data Exchange Platform

Marc Burian, Matr.-No. 344300

Study Program: Bachelor Informatik
September 27, 2017

Supervisor: Prof. Dr. Stefan Decker
Chair of Computer Science 5

Advisor: Benjamin Heitmann, PHD
Chair of Computer Science 5
Abstract

The aim of this thesis is to provide an overview about the feasibility of the Personal Data Store and the Attribute-based Encryption technologies in context with a Data Exchange Platform. Both technologies have achieved a pleasing degree of maturity today, but applications that implement them are rare. For this reason, we will use both approaches in parallel to implement a Data Exchange Platform with respect to conventional privacy and security aspects.

This thesis first examines the case of a Data Exchange Platform and then turns to relevant Big Data technologies that research brought in the past. As mentioned, two of these approaches will be investigated closer and implemented in parallel to provide a framework that can be used to compare them.

The resulting application is used to draw a practical conclusion as to which of either approach is more feasible for our purpose. In addition, it will also identify that today’s solutions for the use of a Data Exchange Platform do not sufficiently use emerging technologies to ensure a fair privacy situation for their user.
Table of Contents

List of Figures ........................................................................................................................................... 4

1 Introduction .............................................................................................................................................. 5
  1.1 Motivation ...................................................................................................................................... 5
  1.2 Thesis Goal .................................................................................................................................... 7
  1.3 Outline .......................................................................................................................................... 7

2 Background ........................................................................................................................................... 8
  2.1 Personal Data Stores ...................................................................................................................... 8
    2.1.1 The Six Main Challenges of Personal Data Store ................................................................... 8
    2.1.2 Securing Personal Data and Big Data ................................................................................... 9
  2.2 Privacy Analyzing and Enhancing Methods .................................................................................. 10
    2.2.1 LINDDUN Reference Model ............................................................................................... 10
    2.2.2 LINDDUN Privacy Enhancement Methodology ................................................................. 11
  2.3 Volume of Data .............................................................................................................................. 12

3 Related Work ....................................................................................................................................... 13
  3.1 Personal Data Stores ..................................................................................................................... 13
    3.1.1 Personal Data Stores: Example (1) ...................................................................................... 13
    3.1.2 Personal Data Stores: Example (2) ...................................................................................... 13
  3.2 Need and Opportunities of Functional Encryption Algorithms .................................................. 14
    3.2.1 Traditional Asymmetric Key-based Encryption ................................................................. 14
    3.2.2 Group Key-based Encryption ............................................................................................... 15
    3.2.3 Attribute-based Encryption .................................................................................................. 15
    3.2.4 Scheme of Ciphertext-Policy AAE ...................................................................................... 17
    3.2.5 Performance of introduced Cryptographical Systems ....................................................... 18
  3.3 Recapture and Overview ................................................................................................................ 19

4 Use Case and Requirements ................................................................................................................ 20
  4.1 Use Case ....................................................................................................................................... 20
  4.2 Requirements .............................................................................................................................. 20
    4.2.1 Functional Requirements .................................................................................................... 20
    4.2.2 Non-Functional Requirements ............................................................................................ 21
    4.2.3 (Non-Functional) Privacy and Security Requirements ..................................................... 21
    4.2.4 Requirements which are not in Scope of this Work ............................................................. 22
  4.3 Recapture and Overview ................................................................................................................ 22

5 Conceptual Approach and a-priori Evaluation .................................................................................. 24
  5.1 Our three Approaches ................................................................................................................... 24
  5.2 Status Quo Approach ................................................................................................................... 25
    5.2.1 Status Quo: Sketch of an implementation ............................................................................. 25
    5.2.2 Status Quo: A-priori Evaluation ............................................................................................ 25
Appendix B

10

Overview

7.4

Recapture and Overview

6.2

Implementation

6.2.6

Implementation of the Evaluation Application

6.1.5

Design of the Data Management Layer and Other Layers

6.1

Software Design

6.1.2

Design of the Application Group Layer

6.1.4

Design of the Configuration Layer

6.1.3

Design of the Cryptographical Layer

6.1.1

Design of the Application Layer

6.2.4

The ABE-based approach

6.2.5

The Pos-based approach

6.2.6

Implementation of the Evaluation Application

6.2.1

Detailed System Overview and Libraries

6.2.2

Data Generation and Data Model

6.2.3

Libraries

6.1

Realization

6

5.5

Recapture and Comparison

5.4

ABE-based Approach

5.4.1

ABE-based Approach: Sketch of Implementation

5.4.2

ABE-based Approach: A-Priori Evaluation

5.3

Pos-based Approach

5.3.1

Pos-based Approach: Sketch of an implementation

5.3.2

Pos-based Approach: A-priori Evaluation

7

4.5

Implementation

4.2.6

Implementation of the Evaluation Application

4.2.5

The ABE-based approach

4.2.4

The Pos-based approach

4.2.3

Implementation of the Evaluation Application

4.2.2

Implementation of the Evaluation Application

4.2.1

Implementation of the Evaluation Application

4.1

Evaluation

4

7.2

Comparison of both approaches

7.2.1

Impact of the Data Size

7.2.2

Impact of Non-Permission Detectability

7.2.3

Impact of Scaling System

7.3

Meaning of the Results

7.4

Recapture and Overview

7.1

Evaluation of the ABE-based approach

7.1.1

Impact of the Number of Attributes

7.1.2

Impact of the Policy Length

7.1.3

Impact of the Policy Height

7.1.4

Plausibility of the Results

3

6

5.1

Recapture and Comparison

5.2

Recapture and Comparison

5.3

Recapture and Comparison

5.4

Recapture and Comparison

5.5

Recapture and Comparison

5

Appendix B

Appendix A

10

Publication Bibliography

Overview

Discussion

5

4

3

- 3 -
List of Figures

Figure 1 Problem Space and Solution Space of Linddun methodology ........................................... - 11 -
Figure 2 Example for a Data flow diagram ....................................................................................... - 11 -
Figure 3 Scheme of a Personal Data Store ......................................................................................... - 13 -
Figure 4 Authorization Request on an Android Marshmallow Build ................................................ - 13 -
Figure 5 RSA-based authentication scheme ........................................................................................ - 15 -
Figure 6 Example for the Policy Public Corruption Office AND (Management-Level > 5 OR Knox
ville OR San Francisco) OR Name: Charlie Eppes) ......................................................................... - 16 -
Figure 7 Sketch of connection of the three phases ............................................................................ - 24 -
Figure 8 Sketch of a trade between a Data Vendor and a Data Customer, mediated by an intermediary party ........................................................................................................................................ - 25 -
Figure 9 Sketch of PDS- based architecture ....................................................................................... - 27 -
Figure 10 Architecture of an ABE-based Data Exchange Platform ................................................ - 29 -
Figure 11 Three-tier architecture ...................................................................................................... - 32 -
Figure 12 Extended Three-layer model ............................................................................................. - 35 -
Figure 13 Overview of Central Instance managing the ABE approach ............................................. - 36 -
Figure 14 Overview of the DataChainSet class .................................................................................. - 37 -
Figure 15 Proceed of a Data Exchange in an Abe-based System ..................................................... - 37 -
Figure 16 Illustration of the Vendor class .......................................................................................... - 38 -
Figure 17 Proceed of a Data Exchange in a PDS-based System ...................................................... - 39 -
Figure 18 Evaluation of ABE-based approach, Number of Attributes (Initialization, Generation, Decryption, Total) ........................................................................................................................................... - 40 -
Figure 19 Abe-based Approach, Evaluation of Policy Length (Initialization, Generation, Decryption, Total)... - 41 -
Figure 20 Abe-based approach, Evaluation of Policy Height (Decryption, Total) ................................. - 41 -
Figure 21 Abe-based approach, Evaluation of Policy Height (Initialization, Generation) .................... - 42 -
Figure 22 Comparison of Data Size (Init+Gen, Decr, Overall) .......................................................... - 43 -
Figure 23 Comparison of Non-Permission Detectability ..................................................................... - 43 -
Figure 24 Comparison of Multiple Access Processes ........................................................................ - 44 -
1 Introduction

1.1 Motivation

Looking at the contribution that computer science and general technical progress have brought, each honest conclusion must be, that a lot of technical components greatly simplify the life of everyone. Van Kleek and O’Hara describe this process as “general[ising] the benefits and enhance the autonomy” [van Kleek and OHara 2014, S. 126] in many contexts where technology is applied. Nevertheless, if from a scientific or from a personal point of view, there also seems to be a threat which is caused by the immense and fast development that the technical environment around us goes through. One typical manifestation of this threat – which many people feel in every day live – is the endangered privacy. Often one wrong click on the internet or one inattentive moment in daily life can be enough to cut deeply in one’s personal privacy. Each photo taken on a mobile phone is just one click away from Google, each step we take is measured by our smartwatches or smartphones. Van Kleek and O’Hara claim the strong statement, that the “storage of personal information is one area where it [meant: technology] has, thus far, been used to power a perverse reversal towards more centralization” [van Kleek and OHara 2014, S. 126].

Agreeing with this pessimistic view or not, both – users and companies – realized this paradigm shift from protected to centralized data – and how this data can be useful for both. Typical reactions which users take to keep the control over their data, such as private data stores as van Kleek and O’Hara present them [van Kleek and OHara 2014] are a defensive answer to the trend. But according to a study of Acquisti and Grossklags [Grossklags and Acquisti 2007, S. 2] there is another motivation for people to consider in managing personal data. In their survey, Acquisti and Grossklags gave people an initial capital of $10. Then parts of the participants were asked if they are interested in selling selected personal data, such as their body-weight or their performance in an IQ-Test, for a certain amount of dollars and other participants where asked if they would spend their initial capital to protect the same personal data, so to prevent it from being publicly revealed. The surprising result was, that – depending on the data that was requested – 80-95% of the surveyed people were willing to sell a single dataset for amounts around $0.25. In the whole survey, only one person wanted to spend $0.25 on protecting their personal data, while in each other case the asked people surveyed this option and kept their initial capital.

Reactions on this circumstance are manifold. Health insurance companies, like BIG direkt gesund, entice their customers with discounts or extra money, if they reveal their healthcare data e.g. Body-Mass Index (BMI) or if they participate in certain sports activities. The payment here is a maximum of €100 per annum, mostly €10 per activity that the user performs1. Another professional instance, that directly pays for personal information is Trendence Institute GmbH that publishes quarterly reports on the popularity of companies from employers’ points of view, based on opinions of employees and other workers. For this purpose, registered users regularly receive an invitation to take an active part in surveys where different questions on employing companies are asked. A survey consists of approximately 30 questions that can be answered in 5 minutes. Usually, the users must describe a

1https://www.big-direkt.de/leistungen/bonusprogramm_bigtionaer/leistungen/vorteilals_bigtionaer/leistungen/massnahmenuebersicht.html
company’s properties or discriminate the company from others by answering targeted questions. The usual payment here is €1 to €2 per 5 minutes.

Beside this few offers where the customers directly trade their data with the customers, most of the online data exchange platforms are third-party provided. Those third-party providers act as data brokers here. Comparable to the application that Trendence Institute GmbH offers, most of these data brokers bring together companies which quickly need a certain amount of opinions to make a business decision. General burdens and benefits of online-surveys have been discussed by Evans and Mathur [Evans and Mathur 2005] from an economical point of view, with the main conclusion, that those studies must be conducted properly with respect to many aspects, like neutral and distorted attributes of Internet population (like upscale, male etc.). Nevertheless, they consider the tools as useful if those aspects are not ignored.

An example of a data broker is The Nielsen Company who maintains a substantial and representative target group and offers the services to ask a single question to this group and then provide their answers live. The service is called Quickquery\(^2\) and the process of sending a question to the users is supported by The Nielsen Company consultants to ensure a useful answer for the interested company. The whole service costs $1,100 per one question and is sent to 2,000 users\(^3\). The Nielsen Company is known for very high standards of quality in tracking consumers’ behavior. A popular example, for instance, are Barcode scanners\(^4\) to track users’ food consumption or Nielsen audience measurement devices for tracking television viewing\(^5\). Additionally, The Nielsen Company is known to pay fair salaries to their representative target group and regularly exchange their target group participants, thus, The Nielsen Company can be assumed to be a high quality and well-suited provider when it comes to questions on users’ consumption behavior. This assumption is also supported by Evans and Mathur who name The Nielsen Company in context with their research on how efficiently online marketing surveys work.

Nevertheless, it is obvious, that such high-quality solutions are not made for mainstream companies. Whether, each interested company can pay an amount of $1,100 for a single question, nor can The Nielsen Company offer each inhabitant of America and Europe to join the group of data vendors. Moreover, The Nielsen Company is known for selecting their target groups autonomous from masses of candidates – A problem that created a breeding ground for pure online data brokers. Examples are Mysurvey, Toluna, SurveyMonkey and much more\(^6\). Instead of measuring user data or selecting their user group with respect to quality standards, those online data brokers offer an open system that allows companies to create surveys which then are provided to their pick ‘n’ mix user group. Waiving those quality standards and providing the whole service on demand, SurveyMonkey offers an unlimited amount of surveys for €400 per year, less than the cost of one single question when choosing The Nielsen Group as a service provider. Although the idea appears to be a good concept to provide a fast and cost-effective overview on users’ opinion, all the savings that these data brokers produce, lead to manifold disadvantages for the data customers and vendors. The data customers have no control on surveying a qualified target group. Instead, the data brokers offer only randomly chosen target groups. This problem is reinforced by the distorted attributes of Internet users, e.g. their young age and their

\(^2\) http://www.nielsen.com/de/de/solutions/capabilities/quickquery.html
\(^3\) http://sites.nielsen.com/meetquickquery/
\(^4\) https://www.nielsen-onlinereg.com/?cpid=1904CE#
\(^6\) A selection can be found here: https://www.bezahlte-umfragen.net/
predominantly male gender, who are not truly representing the general population [Evans and Mathur 2005, S. 8].

Disadvantages on the participant’s side are even more relevant. Answering the surveys from registered accounts enables the receiver to create a detailed user profile. Moreover, due to the online architecture, the user has no control over the data after the submission process, so that the third-party data broker can easily access and analyze the information if he wishes to do so. A third point of criticism is, that the user cannot store his data so that he potentially has to answer the same question more often than once. From a superficial economical point of view, those online systems are not yet mature and a superficial view on privacy aspects is even enough to attest deep privacy threats.

1.2 Thesis Goal

The aim of this thesis is to face the situation stated in the motivation and provide an application that can be used as a framework for an online Data Exchange Platform with enhanced economical and privacy aspects which may be controlled by but is not accessible to, a third-party data broker. The Data Exchange Platform offers services for data vendors who can store and offer their datasets in a protected and trustworthy manner and services for data customers who can access certain datasets if they can meet the access restrictions. To simplify the distinction between both parties, we will assume a data vendor to be male (“he”) and data customer to be female (“she”), while data brokers are neutral (“it”).

As the title of this thesis states, we focus on the feasibility of an Attribute-based Encryption-based (ABE-based) system for our use case, but we will derive this and other solutions logically, prior to comparing them to one another and subsequently discuss the feasibility of the sufficient solutions. To understand which challenges we must face in the context of a Data Exchange Platform, we will discuss related literature on privacy and cryptography induced challenges in relation to Big Data. We will then derive the requirements for a Data Exchange Platform from the identified challenges These theoretical considerations then are supplemented by a practical application. The application will implement two different approaches for a Data Exchange Platform with enhanced privacy and functional aspects, but still based on the main idea of easy accessibility for users and customers so that data queries can be done fast. In conclusion, we will compare the framework with our requirements and from a functional point of view and discuss our findings on the feasibility of the implemented approaches and in specific the ABE-based approach.

1.3 Outline

The remainder of this thesis is structured as followed: Chapter 2 contains background information on security and privacy challenges in big data context and furthermore demonstrates, which volume of data we must assume for a Data Exchange Platform. Chapter 3 complements related approaches that deal with the compromising of certain challenges from the previous chapter, so-called privacy enhancement tools (PETs). In Chapter 4 we will rearrange the gained knowledge to explore, which requirements our implementation of Data Exchange Platform must meet. In Chapter 5 we derive two approaches that are applicable to provide a Data Exchange Platform. Chapter 6 sketches the design and the realization of our implementation. Chapter 7 describes the evaluation and presents the achieved results. In Chapter 8 further, open points are discussed, while Chapter 9 gives an overview of the findings of this work.
2 Background

This chapter gives an overview of basic research and general challenges that the targeted application may be faced with. Section 2.1 introduces the general idea of Personal Data Stores as a concept to store data in a protected way and describes the challenges that partially accessible data, as required for a Data Exchange Platform, must meet. Section 2.2 explains a model to analyze privacy threats and Section 2.3, lastly, contributes assumptions on the data volume and size that we make for our Data Exchange Platform.

2.1 Personal Data Stores

Personal Data Store (PDS) is a scientific term that is generally used for a limited area where personal data is stored. Research in this field is motivated in discussions on finding a compromise between having the best possible user experience and enjoying full privacy on the internet or in (web-)applications. A PDS supports this idea by helping users to keep their data in a personal environment and then individually select which applications they allow to access in this environment. Since the paradigm of retrieving control over personal information was one of the main criticisms on online data brokers in the motivation, the PDS approach is a good starting point, for research in finding a feasible application for Data Exchange Platform that offers fair privacy conditions.

2.1.1 The Six Main Challenges of Personal Data Store

Van Kleek and O’Hara (2014) specified six main challenges of Personal Data Stores. They represent the general goals of a PDS, without excluding each other. However, the challenges contribute a good indication factor for a well-implemented PDS generally.

The first challenge a data store must face is called “longitudinal keeping”, which describes the system robustness to failures and easy migration between different devices so that the data cannot get lost in the long-term. The second challenge is “easy access for everyone” to ensure, that also people who have few experiences with the internet or privacy threats can use the PDS to secure their data – if they have no prior experience with data management and its security burdens. This also includes understandable security aspects and access management. The third challenge is called the “fundamental change” that describes the general situation in which the power over the data withdraws from external applications and is transmitted to the user. Especially, when the data is stored locally this means many new burdens: The data must be protected from illicit changes caused by its owner or by the data broker. Therefore, protected, the data must be stored securely and encrypted, so the administrative role of the data broker must be automated. The fourth challenge is “complying law and other requirements” which describes, that the implementation must meet legal requirements, such as privacy and security laws. The fifth challenge is “monitoring prevention”. This means, that only the use of a well-suited data storage, does not prevent any application from monitoring the user’s behavior and creating a so-called reduced profile. Additionally, this means, that the data which a personal data store offers must not help a company to create a detailed profile of a user when combined with monitored data. A possibility to meet this requirement are additional privacy-enhancement tools, but generally spoken, this fifth challenge is a general and additional aspect that cannot be met by the PDS itself.
To understand, how powerful only monitoring software is, one can refer to www.crunchbase.com\(^7\), where profiles of celebrities are generated, based on their online activities. The last challenge is "future reliability", which is attributable to the fact, that data and data structures changed a lot in the past, and technologies like augmented reality glasses give the hint, that this trend shows no sign of abating. The claim of this challenge is to ensure, that once a PDS is set up for one user, he can use it for a lifetime – if he so wishes – and stays unaffected from further technological changes.

2.1.2 Securing Personal Data and Big Data

Mayer-Schönberg and Cukier sum the term Big Data up as follows: “There is no rigorous definition of big data. Initially, the idea was that the volume of information had grown so large that the quantity being examined no longer fits into the memory that computers use for processing, so engineers needed to revamp the tools they used for analyzing it all.” [Mayer-Schönberger and Cukier 2013, S. 6]. And this situation also applies to security aspects. While local data or at least manageable sized data can be protected by various encryption concepts, many conventional methods do not scale in the context of Big Data.

Rajan et al. [Rajan et al. 2013, S. 1–11] classified the diverse issues that appear in that context and reduced them to the ten greatest privacy challenges in context to Big Data. To derive requirements for our application later, we want to take the findings as a knowledge base. Rajan et al. are more focused on listing general weaknesses that be considered, than suggesting practical solutions for those issues.

Secure Computations in Distributed Programming Frameworks: This aspect concerns the architecture and may occur for example when using a distributed MapReduce framework on a Big Data structure. An untrusted mapper could insert incorrect wrong results into the process and distort the result of the Reduce phase’s aggregation.

1. Security Best Practices for Non-Relational Databases: This aspect describes problems that occur in the context with Non-relational data stores as they are often used in Big Data context. For example, solutions that avoid SQL injection for such databases are not considered to be sufficiently mature.

2. Secure Data Storage and Transactions Logs: The third aspect describes failures when switching between different (hardware-)tiers or rather switching the storage medium which can cause several issues. What we understand as Big Data, usually causes multiple (parallel) transaction processes, which aggravates the problem.

3. End-Point Input Validation/Filtering: The fourth challenge, describes the problem of discriminating valid data from data that was falsified on purpose. The problem is for example caused by so-called ‘bring your own device’ (BYOD) services, which allows users to generate data on their personal measurement devices. Those devices can hardly be prevented from being manipulated.

4. Real-time Security/Compliance Monitoring: This challenge mainly describes the problem, that data with huge volume cannot be completely monitored. This makes it difficult to detect attacks on parts of the data, just when the attack is happening.

---

\(^7\) https://www.crunchbase.com/person/donald-trump#/entity
5. **Scalable and Composable Privacy-Preserving Data Mining and Analytics:** This challenge is induced by the data mining activities of many online companies. They track and store the information they can gain by their users and use them for analytics. If these comprehensive analytics are not anonymized reliable, it is possible to draw conclusions on the individual users behind the analytics which can massively injure the privacy of the users.

6. **Cryptographically Enforced Access and Control and Secure Communication:** This aspect describes the challenge of maintaining end-to-end encryption and access control with traditional methods. These traditional methods are not intended to work in a scaling Big Data environment. Rajan et al. propose attribute-based encryption (ABE) as a solution. We will consider this approach later in detail.

7. **Granular Access Control:** As completion of aspect number 7, this aspect describes the concept to use access rights (or so-called roles). If the access levels are too coarse-grained, a system administrator could tend to protect some data overcautiously in respect to laws or provisions. On the other hand, too many access levels may make the system too complex or even inefficient.

8. **Granular Audits:** This describes the issue of interpreting the data to understand previous processes. Due to the large amount of data, it is complex to find relevant information when an audit is carried out.

9. **Data Provenance:** This challenge generally describes the problem that metadata grows proportional when the core data grows.

### 2.2 Privacy Analyzing and Enhancing Methods

The motivation already indicated some privacy threats in common online *Data Exchange Platform*. To understand which privacy threats generally are distinguished and which enhancing methods exist, we will present the LINDUN model from Deng et al. [Deng et al. 2011] and Pfitzmann et al. [Pfitzmann and Hansen 2010]. The model is related to well-known security models, such as Microsoft’s STRIDE [Microsoft Corporation 2005] model, but it has a stronger focus on privacy aspects.

#### 2.2.1 LINDUN Reference Model

Each letter in LINDUN represents and stands for one privacy threat that a software application that deals with user data must face. The threats can be divided into *hard privacy* and *soft privacy* threats.

*The hard privacy threats refer to the general data minimization problem and should allow users to profit from the benefits of a software while transmitting as little data as possible. The threats that this privacy goals bases on are Linkability, Identifiability, Non-Repudiation, Detectability and Disclosure of Information.* To understand the measures that can be taken against these issues, we will describe the properties that a software must have to avoid that an attacker can use one of the hard privacy threads to take a measure to injure a user’s privacy. Unlinkability can be achieved by encapsulating the data from one another in a way, that links between two or more items of interest (IIOIs) are hidden so that two sets of information do not offer the possibility to generate any additional dependency information. Anonymity – the measure against Identifiability – means that an attacker cannot identify the user that the data belongs to. Pseudonymity describes the possibility to use different identities – or pseudonyms – for different purposes. Plausible deniability prevents users from being detected based on the presence of an act. If the privacy goal is fulfilled, an attacker cannot prove that a user knows that he did (or did not) perform an action. Detectability is faced by hiding information on the existence of an
IOI so that an attacker cannot detect if an IOI exists or not. The last aspect, Disclosure of Information, is faced by controlling the access to data and keep it away from unauthorized people, for example by end-to-end encryption when transmitting it.

The last two privacy threats in the LINDDUN model, Content Unawareness and Policy and Consent Non-Compliance, belong to the field of soft privacy threats, which must be considered for cases where a user decides to transmit his data on purpose. The aspects support the user examining if the policies that he attached to the data are met by the receiver. The first threat of unawareness can be faced by letting the user know what exactly happens to his data. Problems in this field usually occur if the user does not know or does not understand consequences of his act or especially consequences of sharing his personal information. This also includes information that a user shared in the past and that he has no further control over. The only chance to avoid Non-Compliance is a strict and consent compliance that all the users must follow. This includes that the system, by design, prevents serious privacy injuries, which for example can be caused by security weaknesses [Deng et al. 2011, S. 7–10].

It should also be noted, that certain privacy threats can induce each other. A very intuitive example is an unprotected channel, which concerns the Disclosure of Information threat in the first instance, but also makes it potentially easier for an attacker to trace back the data to its origin and injure the Linkability threat.

2.2.2 LINDDUN Privacy Enhancement Methodology

LINDDUN as defined by Deng et al. [Deng et al. 2011] is a methodology for questioning and improving the privacy aspects of a certain software-intensive system. While the improvement part (cp. Figure 1 Solution Space) is not suitable for us, since it requires an existing solution, we can use the Problem Space section to plan our solution.

![Figure 1 Problem Space and Solution Space of LINDDUN methodology](image)

The first process – Defining DFD (Data flow diagram) – consists of the high-level objective to get a clear idea of the data flow, with the focus on the acting parties. Equivalent processes in software planning are for example UML modeling or STRIDE threat modeling process [Microsoft Corporation 2005], all with the main objective to understand the sought data and communication flow on a high level. An example for a DFD can be found in Figure 2, that shows a very simple flow diagram for a conceptual social network. During the second process, Map privacy threats to DFD elements, the different states and flows of the DFD are mapped to the LINDDUN privacy threats. For each threat, Deng et al. contribute patterns that represent the problem space which allows a systematical identification. For instance, the pattern for the Linkability threat lists the problems of different visible user IDs, such as session ID or identification via IP and a not full
protection of the data [Deng et al. 2011, S. 7–10]. We will, in general, use the LINDDUN model to derive extended privacy requirements.

2.3 Volume of Data

Formulating requirements and planning an application requires a minimum of information about the data volume that we are talking about. For this purpose, we want to discuss which data volume we should assume for a Data Exchange Platform. We assume that data customers are interested in a representative number of answers so that, if every data vendor comes from the same country, the data customer must survey enough vendors to represent 100 million people (that is sufficient for all European states and only 7 states in the world have a significantly higher population). Further statistical assumptions on representativeness are made by Lippe et al. [Lippe and Kladroba 2002].

Taking the described population as a basis, we can assume that the number of surveyed datasets must be representative for a population of up to \( N = 100 \) million respondents. A very common confidential level is 99%, meaning that a survey result only has a chance of 1% to differ from the reality more than an error parameter \( e \). This error parameter \( e \) usually is determined between 3% and 5%, we choose \( e = 3\% \) for a better fault tolerance. The formula that we can apply here is

\[
\frac{z^2 \times 0.5(1-0.5)}{e^2} \div (1 + \frac{(z^2 \times 0.5(1-0.5))}{e^2N}).
\]

The \( z \) parameter bases on the assumed error distribution, which is modeled by a Gaussian distribution for deviation. From a standard normal table\(^8\), we can derive the value 2.6 which supplies 1878 respondents for our assumptions. This, for instance, corresponds with the German political polls of Insa Consulere GmbH\(^9\) which weekly queries 2,000 people for their political opinion.

Except for the number of datasets, also the expected dataset volume is interesting. When talking about the exchange of personal data, even a very small dataset might be of interest. Examples here may be the year of birth, salary, preferred party, nationality, favorite movie genre, preferred dinner restaurant, weight, IQ etc. All this data can be represented as char-sequences, so-called Strings, with less than 100 letters. A letter is usually stored in UTF-8 or ASCII Standard, where one letter requires 7-8 bit to be represented so that 100 letters have a size of 700-800bit. Use cases, where documents or pictures are taken as instances for personal data may require a higher number of bits being transferred. A 1080p image, for example, contains \( 1920 \times 1080 = 2073600 \) pixels, where each pixel at most has a Red, Green and Blue (RGB) component, each of 8 bits (= 1 byte) size, so that one 1080p picture may have a size of up to \( 2073600 \times 3 \text{ byte} = 6220800 \text{ byte} \approx 5.93 \text{ Mbyte} \). Due to advantages that compression technology offers, this size can usually be reduced to less than 1 \text{ Mbyte}.

We will take these findings into account in our evaluation later unless meaningful results can already be derived from smaller values. Thus, we will consider systems with approximately 2,000 users (1,878) who have datasets of a size in the range of a few hundred kilobytes.

---

\(^8\) http://math.arizona.edu/~rsims/ma464/standardnormaltable.pdf  
\(^9\) http://www.wahlrecht.de/umfragen/insa.htm
3 Related Work

This chapter gives a basic overview of the cryptographical and conceptual solutions that our Data Exchange Platform will be based upon. Section 3.1 gives an overview of existing implementations of Personal Data Stores, which were already introduced in the previous chapter. Section 3.2 introduces traditional cryptographical algorithms and novel cryptographical algorithms that implement functional aspects on an algorithm level. These algorithms are supposed to ensure the security of our Data Exchange Platform. In Section 3.3, the results are recaptured.

3.1 Personal Data Stores

Personal Data Stores (PDS) were introduced in the previous chapter and are now examined in more detail. The two upcoming subsections each present an example of the implementation of a system that is based on the general PDS approach.

3.1.1 Personal Data Stores: Example (1)

An example for an OpenPDS\(^\text{10}\), provided by de Montjoye et al. [Montjoye et al. 2014], is an application that is based on the purpose to store datasets and automatically approve external parties access to certain parts of the data, based on the data vendors adjustments. A common example, given on the website of the Massachusetts Institute of Technology, is the problem of finding the next song in a music playlist\(^\text{11}\).

Via OpenPDS, a user can simply allow his music applications to access data about his personal taste in music, for example, the songs that the user heard previously or the songs he heard most often. To allow multiple applications access to this kind of personal data, it is sufficient to generate the data uniquely in a database and provide access to the data to differing applications. For that purpose, the user may provide an interface to his database, which an application can use to access the data, after proceeding a credible authentication process.

3.1.2 Personal Data Stores: Example (2)

Beside the OpenPDS implementation, the mobile operating system Android (version 6.0 Marshmallow and higher) implements a similar approach. An analysis of this method was presented by Andriotis et al. [Andriotis et al. 2016]. The Marshmallow Upgrade brought an authorization management environment to Android which has a concept that is similar to the PDS approach. Whenever an application wants to perform a certain action that is provided by Android itself or that needs data that is stored on the smartphone, the user receives an alert window (cp. Figure 4) with a request where he can actively decide whether the application can

\(^{10}\) http://openpds.media.mit.edu/

\(^{11}\) http://news.mit.edu/2014/own-your-own-data-0709
use the need resources or not. This is a typical case, where the user may reveal a part of his privacy, but receives more functionalities for it. An example, where those alerts and access right management is used is WhatsApp\textsuperscript{12}. The core application can be complemented by access to the contact database, sharing pictures from the camera and/or gallery or allowing the use of the microphone to send audio messages.

### 3.2 Need and Opportunities of Functional Encryption Algorithms

Similar to the framework that the PDS concept offers, functional encryption algorithms offer possibilities to build a secure system where users can determine the accessibility of their datasets. Against the PDS approach, the access functionality is implemented on the cryptographical level. To provide an idea of these solutions, we want to introduce traditional, non-functional method for this purpose in Section 3.2.1. Section 3.2.2 presents a method to use normal keys as the representation of roles, while Section 3.2.3 and 3.2.4 come across with a very novel approach in the field of functional encryption algorithms which even enables complex access conditions, straight on to the cryptographical level. In Section 3.2.5 we present former knowledge on the performance of the introduced approaches.

#### 3.2.1 Traditional Asymmetric Key-based Encryption

A simple approach for traditional key-based encryption, that enables users to communicate with each other in a secure way is the use of an asymmetric cryptosystem. The first – and still most common – of these cryptosystems is RSA, which we want to consider as a traditional public-private-key encryption system. RSA today is defined in RFC 3447\textsuperscript{13} and since then there have been many contributions to RSA-based encryption and authentication, as for example by Seifert [Seifert 2005]. The concept of RSA is choosing two prime numbers $p$ and $q$ and computing $n = pq$ and $\varphi(n) = (p - 1)(q - 1)$, which both are multiplications that can proceed efficiently. A public key can now be found by choosing $a \in [1, \varphi(n)]$, where $\gcd(a, \varphi(n)) = 1$. We can publish $K_{\text{pub}} = (a, n)$ then. As private key, we use $b \in [1, \varphi(n)]$, where $ab \equiv 1 \pmod{\varphi(n)}$. Due to the efficiency of the multiplication, the private key can be found out, by trying all viable options, which can be done in an acceptable time. Both keys ($K_{\text{priv}} = b, K_{\text{pub}} = (a, n)$) then work as inverse element of each other. We want to remark, that knowing the public tuple $(a, n)$ is not sufficient to compute $K_{\text{priv}} = b$, which requires decomposing $n$ into its two prime factors (to compute $\varphi(n)$) which we know as a NP-complete problem. If the generating party knows ($K_{\text{priv}} = b, K_{\text{pub}} = (a, n)$) and a sending party knows $K_{\text{pub}} = (a, n)$ they can communicate securely as follows: A message $m$ can be encrypted by using the public key $c(m) = m^a \pmod{n}$, where $c(m)$ is the ciphertext of $m$. Without knowing the inverse of $a$, the ciphertext cannot be decrypted. The generating party is the only party being able to do so, therefore it calculates $m(c(m)) = (m^a \pmod{n})^b \pmod{n} = m$. It is also notable, that this process can be performed inverted, so it holds that $m = (m^a \pmod{n})^b \pmod{n} = (m^b \pmod{n})^a \pmod{n}$.

We will later evaluate the feasibility of such a traditional encryption-based system as a solution for a Data Exchange Platform. However, the RSA scheme does not only offer a reliable asymmetric end-to-end encryption, but also a reliable solution in the field of safe authentication. We have already seen, that authentication is an important aspect for PDS-based systems (Section 3.1.1). The scheme of a RSA-
based authentication proceed is designed as follows: Alice publishes an authentication token. Whenever she wants to authenticate herself to another party (Bob) she proceeds as follows: She takes her token and encrypts it using her private key – and the result is encrypted with Bob’s public key. She sends the token to Bob. No one else but Bob can decrypt the first shell since only he has his private key. To be sure now that Alice has sent the message, he uses her public key and decrypts the result again. If he finds the token, that Alice had published before, he can be sure, that he is communicating with her. This scheme is sketched in Figure 5 and was, for example, explained and analyzed by Seifert [Seifert 2005].

3.2.2 Group Key-based Encryption

A well-explored approach in the field of controlled access management is the use of role-based access control (RBAC) as provided by Ferraiolo et al. [Ferraiolo et al. 1995]. RBAC is known to be a more key economical solution than the RSA approach from Section 3.2.1 and is motivated in the fact that in a traditional public-private-key system, a very high user number causes an indomitable growth in key management effort. RBAC instead groups users and gives them role keys, assuming, that for the users with the same roles, the same access rights apply. Therefore, the datasets are attached to user roles, that describe which level of access a user must have to decrypt the considered data. Each user owns a private key for each of his roles, which he can use to decrypt data which was encrypted with the associated encryption key.

Applied to our application, a use case for this approach could be as follows: A data vendor who regularly measures his health data wants to store and sell it based on a personal data store. He encrypts his data with a role key which is associated with the role “scientist”. Doing so, the user avoids, that his personal data could be used in industrial or profit orientated contexts. Even if this scheme contributes the basics for a flexible system it is, due to multiple attack schemes, virtually impossible to use these roles to create fine-grained policies. A scheme of nested encryptions (meaning, that a data set is encrypted twice, with two different keys) could, for example, be circumvented by two disjointed users, where each of them has one of the required roles. This kind of attack is called collusion-attack and we will see how to face it – and even improve the scheme – in the next section. Even though through its limits such a system is not feasible, we will keep the idea of RBAC in mind, since the next section which describes a much more mature approach, is a development based on this scheme.

3.2.3 Attribute-based Encryption

An alternative, more complex, more policy orientated and collusion-attack avoiding approach was originally presented by Goyal et al. [Goyal et al. 2006] and Bethencourt et al. [Bethencourt et al. 2007]. The so-called Attribute-based Encryption (A\textsubscript{BE}) system relies on attributes, represented by private keys, that the data customer can use to proof that she has got certain properties. There are two different types of Attribute-based Encryption systems: Key-policy Attribute-based Encryption (KP-A\textsubscript{BE}) which was introduced by Goyal et al. [Goyal et al. 2006] and Ciphertext-policy Attribute-based Encryption (CP-A\textsubscript{BE}) which was introduced by Bethencourt et al. [Bethencourt et al. 2007].

In a KP-A\textsubscript{BE} system, the data customer retrieves datasets which are attached with certain attributes. The data is protected by a so-called access policy. An illustrating example is a movie platform, where a
data customer owns a certain access policy key like _LOW BUDGET PRODUCTION OR (THRILLER AND AGE RATING 18+)_. The data vendor then broadcasts his movies attached with describing attributes, such as \{ROMANCE, SITCOM, AGE RATING 12+\}. If a data customer’s policy is fulfilled by these attributes, she will be able to decrypt the movie – otherwise not. This also explains the prefix Kp \(\text{(Key-Policy)}\), which represents the fact that the user’s decryption keys are tied to policies which determine the users’ access limits. [Goyal et al. 2006]

In contrast to that, CP-ABE represents a system where the ciphertexts are attached with policies. In this case, the data vendor can choose an individual policy for his datasets, as for example _Scientific OR (Non-Profit AND Environmentally Responsible)_, where the attributes represent properties of the data customers. Each customer who has a set of attributes \(S\) which fulfills this policy (for example (Scientific)) can encrypt the underlying data. This also explains the prefix Cr that stands for _Ciphertext-Policy_. By construction, CP-ABE avoids different parties to share their attributes with each other to proceed collusion attacks. We will take a closer look at this in Section 3.2.4 [Bethencourt et al. 2007].

If we compare both types of Attribute-based encryption, that is CP- and KP-ABE, we note, that CP-ABE is the one that is more suitable for our use case of constructing a _Data Exchange Platform_. The high-level application which we can construct with CP-ABE would be a database, where each dataset is accessible if and only if the accessing data customer has the matching attributes to fulfill the policy. KP-ABE, therefore, does not offer a very fine-grained possibility for a vendor to protect his data, because he can only choose a set of attributes to protect a dataset, but he has no control on the composition of the customer’s policy, which practically would make such a scheme too coarse-grained for our use case.

Before we look at the explicit schema of the CP-ABE algorithm, we want to evaluate the representation of the policies that are attached with a dataset. The policies are constructed as access trees \(\mathcal{T}\). The tree can be divided into leaves and non-leaves. Non-leaves represent a threshold gate, that is a value \(\text{num}_x\) and the number of successors \(\text{(children)}\) \(k_x\), where \(x\) is the node. To guarantee the satisfiability it must hold that \(0 < k_x < \text{num}_x\). \(k_x = 1\) represents an _Or_-gate and \(k_x = \text{num}_x\) represents an _And_-gate. Before defining the _Satisfaction_ of a policy, we define a few more functions: _parent_ \((x)\) is the predecessor of a node \(x\), _att_ \((x)\) is the attribute associated with a leaf, so it is only defined if \(x\) is a leaf.

Bethencourt et al. define Satisfaction of an access tree as follows: “Let \(\mathcal{T}\) be an access tree with root \(r\). Denote by \(\mathcal{T}_x\) the subtree of \(\mathcal{T}\) rooted at the node \(x\). Hence \(\mathcal{T}\) is the same as \(\mathcal{T}_r\). If a set of attributes \(\gamma\) satisfies the access tree \(\mathcal{T}_x\), we denote it as \(\mathcal{T}_x(\gamma) = 1\). We compute \(\mathcal{T}_x(\gamma)\) recursively as follows: If \(x\) is a non-leaf node, evaluate \(\mathcal{T}_x(y)\) for all children \(x'\) of node \(x\). \(\mathcal{T}_x(y)\) returns if and only if at least \(k_x\) children return 1. If \(x\) is a leaf node, then \(\mathcal{T}_x(y)\) returns 1 if and only if \(\text{att}(x) \in \gamma\).” [Bethencourt et al. 2007, S. 326] An example of such a policy is given in Figure 6.
3.2.4 Scheme of Ciphertext-Policy ABE

We want to summarize the schema that Bethencourt et al. [Bethencourt et al. 2007] provided to get an idea of the encryption system’s functionality and features. The keys that a CP-ABE system requires are constructed in a pairing-based cryptographic system. Therefore, a bilinear map $e: \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ is needed, where \( \mathbb{G}_0 \) is a bilinear group of prime order \( p \) with a generator \( g \) (that is \( \mathbb{Z}_p^* \)). The main properties of \( e \) are bilinearity (that is \( e(aP, bQ) = e(P, Q)^{ab} \)) and non-degeneracy (that is \( e(g, g) \neq 1 \)). A hash function \( H: \{0,1\}^* \rightarrow \mathbb{G}_0 \) is a comfortable possibility to hash the (binary encoded) name of an attribute name to a value from \( \mathbb{G}_0 \). Upon these structures, we can build the main functioning that CP-ABE uses, that is the functioning \textbf{Setup}, \textbf{Encrypt}, \textbf{KeyGen}, \textbf{Delegate} and \textbf{Decrypt}.

\textbf{Setup} is a linear procedure, where the bilinear group \( \mathbb{G}_0 \) of prime order \( p \) with generator \( g \). Two random exponents \( \alpha, \beta \) then form the public key \( PK \) and the master key \( MK \), that is

\[
PK = (\mathbb{G}_0, g, h = g^\beta, f = g^{1/\beta}, e(g, g)^\alpha)
MK = (\beta, g^\alpha).
\]

\textbf{Setup} is used once when the central instance sets up the system.

\textbf{Encrypt}(\( PK, M, \mathcal{T} \)) is the process that encrypts a message \( M \) under a policy represented by the access tree \( \mathcal{T} \), only by using the public key \( PK \). The encryption works along the access tree \( \mathcal{T} \), which is arranged as follows: A polynomial \( q_x \) is chosen for each node \( x \). These polynomials are chosen to represent the access trees, which works as follows: Starting from the root \( r \), for each node \( x \) the degree \( d_x \) of the polynomial \( q_x \) is set on \( d_x = k_x - 1 \), that is, one less than the threshold. After choosing the degree \( d_x \), we need \( d_x + 1 \) points to define the polynomials clearly. This is done inductively, as follows: For \( r \) a random value \( s \in \mathbb{Z}_p \) is chosen and \( q_r := s \cdot d_r \) other, random, values complete the polynomial \( q_r \). For \( x \) (a node but not the root) we choose \( q_x(0) := q_{parent(x)}(index(x)) \) and \( d_x \) other, random, values to complete the polynomial. The complete ciphertext then formally is

\[
CT = (\mathcal{T}, \tilde{C}_i := Me(g, g)^{as}, C_i := h^s, \forall y \in Y: C_y := g^{q_y(0)}, C'_y := H(att(y))^{q_y(0)})
\]

where \( Y \) is the set of all nodes that occur in \( \mathcal{T} \). Encrypt can be proceeded by each instance multiple times.

\textbf{KeyGen}(\( MK, S \)) is the procedure, that for a set of attributes \( S \) outputs a key that certifies a customer to have that set of attributes. A random \( r \in \mathbb{Z}_p \) and random \( r_j \in \mathbb{Z}_p \) for each attribute \( j \in S \) are chosen and form the key

\[
SK = (D := g^{\alpha + r/\beta}, \forall j \in S: D_j = g^r \cdot H(j)^{r_j}, D^{r_j} = g^{r_j}).
\]

KeyGen is proceeded uniquely by the central instance.

\textbf{Delegate}(\( SK, \tilde{S} \)) is the delegation algorithm that creates a secret key for an attribute subset \( \tilde{S} \subseteq S \), without using \( MK \). The secret key, therefore, is re-randomized, so that it is not different to a key that is provided by the central authority. Therefore, random values \( \tilde{r} \in \mathbb{Z}_p \) and a random value \( \tilde{r}_j \in \mathbb{Z}_p \) for each attribute \( j \in \tilde{S} \) are chosen so that the key for the subset \( \tilde{S} \) can be defined as follows:

\[
SK = (D := Df^{\tilde{r}}, \forall k \in \tilde{S}: D_k = D_kg^{\tilde{r}}H(k)^{\tilde{r}_k}, D^{\tilde{r}_k} = D^{\tilde{r}}_k g^{\tilde{r}_k})
\]

Delegate can be carried out by each instance and as often as desired.
Decrypt(CT, SK) describes the process where a vendor decrypts a ciphertext with his attribute set S, represented by his secret key SK. As the encryption algorithm, decrypt is a recursive algorithm, this time beginning at the leaves instead of the root. If x is a leaf node, then i := att(x) and

\[ \text{DecryptNode}(CT, SK, x) = \frac{e(D_x, C_x)}{e(b_i, C_i)} = \frac{e(g^{r_i H(i)^r_i H^i(0)} g^{q_i(0)})}{e(g^{r_i H(i)^r_i H^i(0)} g^{q_i(0)})} = e(g, g)^{r_i q_i(0)}, \text{if } i \in S \]

\[ \perp, \text{if } i \notin S. \]

With this function as a starting point, we can define the decryption for an arbitrary node x. This is defined as follows: For all children z of x call DecryptNode(CT, SK, z) and store the output as Fz. Let Sz be an arbitrary kz-sized of children z, so that \( \forall z \in S_x: F_z \neq \perp \). That is, at least \( k_x \) children (the threshold value) must be satisfied, so that the next computation is successful and that is what makes up the encryption. If such a set Sz does not exist, we define DecryptNode(CT, SK, x) := \( \perp \), but if it does, we define

\[ \text{DecryptNode}(CT, SK, x) := \prod_{z \in S_x} F_z^{\Delta_y S_y(0)}, \text{where } i = \text{index}(z), S_x = \{\text{index}(z): z \in S_x\} \]

\[ = \prod_{z \in S_x} (e(g, g)^{r_i q_i(0)})^{\Delta_y S_y(0)}, \text{by construction of } F_z \]

\[ = \prod_{z \in S_x} (e(g, g)^{r_i q_{\text{parent}(x)}(\text{index}(z))})^{\Delta_y S_y(0)}, \text{by construction of Encryption} \]

\[ = \prod_{z \in S_x} (e(g, g)^{r_i q_i(0)})^{\Delta_y S_y(0)}, \text{by construction and bilinearity} \]

\[ = e(g, g)^{r_i q_y(0)}, \text{by using polynomial interpolation} \]

where \( \Delta_y S_y(x) = \prod_{j \in S_j, j \neq i} \frac{x - j}{x - i} \).

The function now allows us to define the decryption algorithm, that is the computation of DecryptNode(CT, SK, r) = e(g, g)^{r_q(0)} = e(g, g)^{r_y} which we can use to compute

\[ M = \tilde{C}/(e(C, D)/A) = \tilde{C}/(e(h^s, g^{(\alpha + r)/\beta}))/e(g, g)^{r_y}. \]

As mentioned, schemes which allow different roles or attributes are faced with so-called collusion attacks. Bethencourt et al. sketch how this scheme prevents collusion attacks as follows: “Collusion attacks won’t help since the blinding value is randomized to the randomness from a particular user’s private key”. [Bethencourt et al. 2007, S. 325] A more detailed proof, however, can be found in Appendix A.

### 3.2.5 Performance of introduced Cryptographical Systems

For the two cryptographical systems that were introduced, we want to present former performance analyses. Table 1 shows the performance (in seconds) of a Java implementation of RSA, as it is recommended for security-sensitive software (that is, using OAEP padding, which means that a clear message m is increased to fit a certain size). The performance study was carried out by Preetha and Nithya [Preetha and Nithya 2013]. For our application, we can anticipate that an asymmetric cryptographic algorithm — if additional encryption is required — offers the best performing solution.
Even though a symmetric approach offers a faster overall encryption, its initial overhead is too significant for small datasets. Thus, the initial secret-sharing phase on its own would have a similar performance to a complete RSA-based data transmission. Another important aspect is that the RSA-based approach can be purely based on broadcasting and therefore does not require direct contact between vendor and customer. This will lead to further improved privacy aspects. Table 1 shows that with an input size of 30kb the highest transfer rate (320 bit/s) is reached.

<table>
<thead>
<tr>
<th>Input Size</th>
<th>Encryption Time (s)</th>
<th>Decryption Time (s)</th>
<th>Total Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>14kb</td>
<td>15</td>
<td>63</td>
<td>78</td>
</tr>
<tr>
<td>24kb</td>
<td>16</td>
<td>76</td>
<td>92</td>
</tr>
<tr>
<td>30kb</td>
<td>15</td>
<td>78</td>
<td>93</td>
</tr>
<tr>
<td>39kb</td>
<td>15</td>
<td>170</td>
<td>185</td>
</tr>
<tr>
<td>62kb</td>
<td>16</td>
<td>190</td>
<td>206</td>
</tr>
</tbody>
</table>

**Table 1** Performance of RSA-OAEP as applied in practice

For the Cp-ABE based processes, Ambrosin et al. [Ambrosin et al. 2015] contributed a performance analysis. In their work, they consider the leverage that a growing number of attributes has on the time performance. Ambrosin et al. followed the intention to show the feasibility for ABE cryptosystems on smartphone devices, so that we can only refer to the benchmark results that were measured on a laptop. In general, there are only few feasibility evaluations of the ABE-based approach, so that we cannot rely on former researches here.

### 3.3 Recapture and Overview

Overall, it is difficult to imagine that traditional encryption algorithms can exist in a performance critical Big Data context. Group key-based encryption is at first promising approach that extends traditional encryption with a functional component to achieve time savings. Attribute-based access control surpasses this approach from a functional point of view. However, we lack knowledge of performance so that we cannot draw a definitive conclusion as to its suitability at the moment.

<table>
<thead>
<tr>
<th>Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
</tr>
<tr>
<td><strong>Policy accuracy</strong></td>
</tr>
<tr>
<td><strong>Needed keys</strong></td>
</tr>
<tr>
<td><strong>Expenses</strong></td>
</tr>
</tbody>
</table>

**Table 2** Comparison of different Access Management Approaches
4 Use Case and Requirements

This chapter presents the use case and lays down the requirements for our Data Exchange Platform. Section 4.1 sums up the use cases which the application will implement. Section 4.2 combines the challenges from Chapter 2 with the use case and derives the requirements that our application must meet. In Section 4.3 we will summarize and recapture our findings.

4.1 Use Case

A Data Exchange Platform, in our concept, is an application that allows vendors to provide enclosed data sets which can be accessible for certain groups of data customers. From a data vendor’s perspective, the Data Exchange Platform will basically operate as an easily accessible and securely protected private database. Data customers can browse through the data vendors’ databases and look out for datasets that seem to be interesting to them. Whenever a data customer finds a dataset that she is interested in, she can send an access request and receives the dataset as a response if she is permitted accessing it. The implementation of this access management will be approach sensitive.

Functional requirements from data vendors point of view: Data vendors can mostly be understood as consumers. Their activity is submitting their data and determining the conditions for a customer accessing it. Data vendors may not have expert knowledge in technologies, nevertheless, they should be able to use the Data Exchange Platform without disadvantages.

Functional requirements from data customers point of view: Data customers can be seen as companies or similar establishments that are interested in analyzing certain data from a chosen target group. They browse through the data vendors’ datasets and request the datasets they are interested in.

4.2 Requirements

The previous information on the main use cases of the application, combined with the general challenges and methodologies from Chapter 2 lead us to the general functional requirements and non-functional requirements for our application. Functional requirements mainly describe the functioning, the Data Exchange Platform must provide for the vendors and customers, while non-functional requirements are superordinate requirements that the system must meet. We also want to distinguish non-functional requirements that affect security and privacy aspects, so that we include non-functional security and privacy requirements as a separate section. Lastly, requirements which are not in scope are considered, to justify, why we will not look after certain challenges, that were identified in Chapter 2.

4.2.1 Functional Requirements

Functioning for Vendors: A data vendor can enter personal information and determine customers who are allowed to access his data. He can rely on the enforcement of his conditions without further control. Additionally, the process of trading his personal data is as efficient as possible, so that the vendor can easily manage his data by spending as few time as possible. To decide, which customers he allows to access his data, he needs any prior knowledge about the customers. That could, for example, be a subscription or a reliable attribute-based access control.
Supporting Customers: A customer who is interested in a certain data domain can filter datasets by their domain and access as much of them as he is permitted to access.

4.2.2 Non-Functional Requirements

Usability and Transparency: A vendor can use the system without spending too much effort on managing his personal data. Once the data and its access conditions are uploaded to the Data Exchange Platform, they do not have to spend additional effort in managing it. Furthermore, to grant an Easy Access even for technical unfamiliar people (cp. Section 4.2.2), the system is transparently and trustworthy, so that the user knows his rights and obligations after providing his personal data, but he can also rely on the enforcement of the policy that he determined in context with his personal data.

Data Distribution and Storage Concept: Many of the previous aspects are affected by the level of data distribution and the storage concept. The more distributed the vendor’s data is, the more we must face mentioned challenges like the points of Secure Computations in distributed programming frameworks (cp. Section 2.1.2) and Real-Time Security / Compliance Monitoring (cp. Section 2.1.2). On the other hand, if we choose a completely centralized architecture, it will be difficult to apply Security best practices for Non-Relational Databases (cp. Section 2.1.2) and grant Longitudinal Storage (cp. Section 2.1.1). We want to look at the security aspects in separately, thus this requirement generally consists of the fulfillment of Longitudinal Storage, Security Best Practices and Real-Time Security / Compliance Monitoring.

Performance and Feasibility: The importance of tracking the feasibility of an application that is applied in a Big Data context, was stated as Fundamental Change (cp. Section 2.1.1), that is enforced by the technical development. Therefore, we must consider performance and especially feasibility aspects. Feasibility may, for example, include concerns about the Future Reliability (cp. Section 2.1.1) of the provided application.

4.2.3 (Non-Functional) Privacy and Security Requirements

Privacy: Fulfillment of privacy requirements is one of the main requirement that our contribution must meet to improve the status quo situation. To check the fulfillment level, we will take the LNDDUN privacy model (cp. Section 2.2.1) as a basis. In the point of Scalable and Composable Privacy-Preserving Data-Mining and Analytics (cp. Section 2.1.2), we have already learned, that compromising a well-performing (Big Data) application with privacy aspects is an important problem. We must consider the privacy-aspect bidirectional here: For data customers and for data vendors. While it is obvious that we talk about the datasets when considering the vendors perspective, the customer also produces information that is worth protecting. That is, for example, the fact that a customer is looking for a certain kind of information in many databases, which may be mission-critical information for her.

For the data vendor, we can adopt the following LNDDUN aspects Linkability, Identifiability, Non-Repudiation, Disclosure of Information and Non-Compliance to our requirements. Regarding our former requirements, the Detectability aspect is incompatible. A data customer cannot find an object if she does not know, that the object she is looking for even exists. Therefore, we can try to avoid Detectability for customers, since a vendor does not have to know, whether a customer is interested in his data or not. The aspect Unawareness is application-sensitive so that we do not want to align our framework by this aspect. For the data customer, we also adapt Linkability, Identifiability, Non-Repudiation, Disclosure of Information and Non-Compliance as a requirement. Additionally, we want to add Detectability, because we may find a solution that allows a customer to remain undetected when accessing vendor’s personal data.
Security and Access Management: From an economical point of view it is important to prevent the stored information from being stolen since once they are publicly released, they lose their value for a potential data customer. It is obvious, that we must face the Secure Storage and Transaction challenge, in combination with Granularly determined Access Conditions (cp. Section 2.1.2) that are Cryptographically Enforced (cp. Section 2.1.2). Moreover, Data Provenance (cp. Section 2.1.2) is an influencing aspect in context with security aspects too. The meta data that we have to consider are keys and other cryptographical structures that are attached to a dataset to ensure the user’s provisions to be enforced. We have to minimize this cryptographical meta data or select a solution which scales appropriately in a Big Data context.

4.2.4 Requirements which are not in Scope of this Work

External Factors: We have some external factors given, that may influence a Data Exchange Platform, especially when applied in practice: That may on the one hand be Local Restriction and Laws (cp. Section 2.1.1), which are out of scope for us, because we do not want to build an application-sensitive system, but only a framework. On the other hand, we have considered challenges like Monitoring Prevention (cp. Section 2.1.1) and the creation of reduced profiles, meaning that data in the platform may also be available from an external source, that concluded an information from monitoring the user’s behavior. This aspect is also out of scope for the Data Exchange Platform, since we do not take worth preserving (of the data) as a challenge for our framework.

Others: The problem of End-Point Input Validation/Filtering (cp. Section 2.1.1) described a malicious data vendor who provides forged data to our system. Inspired by The Nielsen Company’s approach, a solution for this may be the use of qualified measurement devices for certain data, which may ensure their correctness. Lastly, Granular Audits (cp. Section 2.1.2) are not in scope of this work, since we decided to focus on Unawareness of the customers interactions which stands in conflict to logging access processes.

4.3 Recapture and Overview

We have now used the background knowledge from Chapter 2 to find challenges that a Data Exchange Platform face. Table 3 systematically summarizes the relationship between challenges and the identified requirements.

<table>
<thead>
<tr>
<th>Researcher</th>
<th>Name of determined challenge</th>
<th>Derived Requirement</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>v. Kleek et</td>
<td>Longitudinal keeping</td>
<td>Data Storage &amp; Distribution Concept</td>
<td>4.2.2</td>
</tr>
<tr>
<td>v. Kleek et</td>
<td>Access for individuals with little experience</td>
<td>Usability and Transparency</td>
<td>4.2.2</td>
</tr>
<tr>
<td>v. Kleek et</td>
<td>General Fundamental change</td>
<td>Performance and Feasibility</td>
<td>4.2.2</td>
</tr>
<tr>
<td>v. Kleek et</td>
<td>Complying with law</td>
<td>Out of Scope</td>
<td>-</td>
</tr>
<tr>
<td>v. Kleek et</td>
<td>Monitoring Prevention</td>
<td>Out of Scope</td>
<td>-</td>
</tr>
<tr>
<td>v. Kleek et</td>
<td>Future Reliability</td>
<td>Performance and Feasibility</td>
<td>4.2.2</td>
</tr>
<tr>
<td>Rajan et al.</td>
<td>Secure computations in distributed programming frameworks</td>
<td>Security</td>
<td>4.2.3</td>
</tr>
<tr>
<td>Rajan et al.</td>
<td>Security best practices for non-relational data stores</td>
<td>Data Storage &amp; Distribution Concept</td>
<td>4.2.2</td>
</tr>
<tr>
<td>Rajan et al.</td>
<td>Secure data storage and transaction logs</td>
<td>Security</td>
<td>4.2.2</td>
</tr>
<tr>
<td>Rajan et al.</td>
<td>End-point input validation/filtering</td>
<td>Out of Scope</td>
<td>-</td>
</tr>
<tr>
<td>Rajan et al.</td>
<td>Real-time security/compliance monitoring</td>
<td>Data Storage &amp; Distribution Concept</td>
<td>4.2.2</td>
</tr>
<tr>
<td>Rajan et al.</td>
<td>Scalable and composable privacy-preserving data mining and analytics</td>
<td>Privacy</td>
<td>4.2.3</td>
</tr>
<tr>
<td>Rajan et al.</td>
<td>Cryptographically enforced access control and secure communication</td>
<td>Security</td>
<td>4.2.3</td>
</tr>
<tr>
<td>Authors</td>
<td>Description</td>
<td>Category</td>
<td>Page</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------</td>
<td>----------------</td>
<td>------</td>
</tr>
<tr>
<td>Rajan et al.</td>
<td>Granular access control</td>
<td>Security</td>
<td>4.2.3</td>
</tr>
<tr>
<td>Rajan et al.</td>
<td>Granular Audits</td>
<td>Out of Scope</td>
<td>-</td>
</tr>
<tr>
<td>Rajan et al.</td>
<td>Data provenance</td>
<td>Security</td>
<td>4.2.3</td>
</tr>
<tr>
<td>Deng et al.</td>
<td>Linkability for Data Vendors</td>
<td>Privacy</td>
<td>4.2.3</td>
</tr>
<tr>
<td>Deng et al.</td>
<td>Identifiability for Data Vendors</td>
<td>Privacy</td>
<td>4.2.3</td>
</tr>
<tr>
<td>Deng et al.</td>
<td>Non-Repudiation for Data Vendors</td>
<td>Privacy</td>
<td>4.2.3</td>
</tr>
<tr>
<td>Deng et al.</td>
<td>Detectability for Data Vendors</td>
<td>Out of Scope</td>
<td>-</td>
</tr>
<tr>
<td>Deng et al.</td>
<td>Disclosure of Information for Data Vendors</td>
<td>Privacy</td>
<td>4.2.3</td>
</tr>
<tr>
<td>Deng et al.</td>
<td>Unawareness for Data Vendors</td>
<td>Out of Scope</td>
<td>-</td>
</tr>
<tr>
<td>Deng et al.</td>
<td>Non-Compliance for Data Vendors</td>
<td>Privacy</td>
<td>4.2.3</td>
</tr>
<tr>
<td>Deng et al.</td>
<td>Linkability for Data Customers</td>
<td>Privacy</td>
<td>4.2.3</td>
</tr>
<tr>
<td>Deng et al.</td>
<td>Identifiability for Data Customers</td>
<td>Privacy</td>
<td>4.2.3</td>
</tr>
<tr>
<td>Deng et al.</td>
<td>Non-Repudiation for Data Customers</td>
<td>Privacy</td>
<td>4.2.3</td>
</tr>
<tr>
<td>Deng et al.</td>
<td>Detectability for Data Customers</td>
<td>Privacy</td>
<td>4.2.3</td>
</tr>
<tr>
<td>Deng et al.</td>
<td>Disclosure of Information for Data Customers</td>
<td>Privacy</td>
<td>4.2.3</td>
</tr>
<tr>
<td>Deng et al.</td>
<td>Unawareness for Data Customers</td>
<td>Out of Scope</td>
<td>-</td>
</tr>
<tr>
<td>Deng et al.</td>
<td>Non-Compliance for Data Customers</td>
<td>Privacy</td>
<td>4.2.3</td>
</tr>
</tbody>
</table>

**Table 3** Listing of all challenges in context with a Personal Data Exchange Platform
5 Conceptual Approach and a-priori Evaluation

This section will bring together the proposed privacy enhancing approaches from Chapter 3 with our use case and requirements. We will present two novel approaches that we consider to be more feasible than the Status Quo approach, which we will justify in the following sections. Section 5.1 introduces the two novel approaches and the Status Quo approach explains their general procedure. Sections 5.2 to 5.4 then outlines the approaches in detail and gives an overview of how well they meet the requirements of Chapter 4. Lastly, Section 5.5 sums up the results in a condensed form.

5.1 Our three Approaches

Our first approach is called Status Quo Approach and describes a straight-forward idea of managing each dataset individually. Vendors only become active, when there is a direct demand for their data. Whenever this happens, they decide whether to ignore or accept the request. If a vendor decides to accept a customer’s request, he generates the appropriate dataset and transmits it in an encrypted way. The second approach is the PDS-based approach that takes up the concept of a Personal Data Store, as van Kleek and O’Hara [van Kleek and OHara 2014] proposed, to present a solution, where adjustable interfaces are used for the communication between vendors and customers. The last approach is the ABE-based approach that is based on the idea to encrypt each dataset with a policy and granting reliable access policy enforcement while data sets are continuously encrypted.

To support the comparability of the approaches, it is useful for us to design common phases for all three approaches. For our application, these phases are Initialization (INIT), Generation/Encryption (GEN) and Decryption (DECR). INIT is meant to be a unique phase that is executed at the initial state of a system, for example, to generate the necessary – but empty – databases. GEN describes a phase where data is generated and encrypted, while DECR describes the customers’ access – and decryption attempts. GEN and DECR are separated logically but may alternate. This is also illustrated in Figure 7 Sketch of connection of the three phases.

The Initialization phase stands for the approach-specific generation of cryptographical and non-cryptographical, mandatory structures. Generation of the private keys may be an example for such a structure, supplying the keys to the vendors may be an example for a method. The Generation/Encryption phase contains the generation of the data, its encryption and the determination of access rights. In the Decryption phase, Customers with adequately access rights can send requests for certain datasets that are transferred in a secure way then. Table 4 presents an overview of the abbreviations we use for the three phases in the three approaches.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Initialization</th>
<th>Generation / Encryption</th>
<th>Decryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABE-based approach</td>
<td>ABE-INIT</td>
<td>ABE-GEN</td>
<td>ABE-DECR</td>
</tr>
<tr>
<td>PDS-based approach</td>
<td>PDS-INIT</td>
<td>PDS-GEN</td>
<td>PDS-DECR</td>
</tr>
<tr>
<td>Status Quo</td>
<td>SQ-INIT</td>
<td>SQ-GEN</td>
<td>SQ-DECR</td>
</tr>
</tbody>
</table>

Table 4 Subdivision of the three approaches in their three phases

Each approach is designed to fulfill at least two requirements: The general functionality, that is, a customer can receive data from a vendor in any way is always provided. Furthermore, the same applies to general security aspects, that is, data is always stored and transmitted in an encrypted way then.
5.2 Status Quo Approach

Implementations of the Status Quo Approach can, for example, be found when considering online survey tools, as they were mentioned in the introduction. The general procedure there is as follows: A data vendor learns that a customer is interested in his data, generates the data (e.g. filling out a survey) and submits it to the customer. The entire process is guided by a mediating party, which ensures a successful and secure transmission. This approach concludes to a very temporary relationship between a vendor and a customer. The communication cannot be continued at a later time, nor can the vendor re-use his data. This also means that cryptographically necessary primitives are only valid for one trade between a vendor and customer.

5.2.1 Status Quo: Sketch of an implementation

The Status Quo approach sketches a privacy-oriented approach that is based on traditional public-private-key schemes and uses a direct vendor-to-customer communication approach in a one-to-one relationship. The data vendor generates his datasets in an application-specific way and encrypts it with a public key from a customer. The customer then can use his appropriate private key to decrypt the data. We do not consider asymmetric crypto systems here since it would require a more directed communication between both parties, which may allow the customer to remember a Vendor, when communication with him at a later point in time. Despite that intuitive definition, we want to give a comprehensibly overview of the functions that the different phases of a Status Quo Approach have.

SQ-INIT: In the Initialization phase of a Status Quo Approach-based system, vendors and customers are matched to proceed a trade. Additionally, each party generates its private and public key.

SQ-GEN: In the Generation phase, the vendor generates his data encrypts it and sends it to a customer.

SQ-DECR: In the Decryption phase, the mediating service forwards the data to the customer who, if necessary, decrypts it. A mediating instance can additionally ensure that an optional financial reward can be transmitted from the customer to the vendor.

Due to the design of this approach, data is never stored for a long period.

5.2.2 Status Quo: A-priori Evaluation

Functional Requirements: This approach does not completely fulfill our functional requirements. While functional aspects concerning the data customer are covered well, the data vendor must accept disadvantages. Firstly, a data vendor has no prior possibility to determine whether he allows a certain customer to access his personal data. He must make his decision repetitively and before every trade. Secondly, systems that are based on the Status Quo Approach, as we have for example seen them in Section 1.1, do not offer a possibility to store data after generation. That induces that the vendor may have to generate a single dataset multiple times – for each customer who is interested.
Usability and Transparency: Usability requirements are not satisfied by this approach. We already noticed that a further recycling of entered data is not possible, which may require users to repetitively enter the same data. Furthermore, current systems usually grant complete anonymity for the data customers which lacks transparency for a vendor who has to decide whether he wants to allow a customer access to a certain information or not without having information on her.

Data Storage & Distribution Concept: The architecture provides a fair solution to handle different issues that may occur in the context of scaling data. The Longitudinal Keeping problem is avoided by avoiding any long-term storage in the system. Security Best Practices for Non-Relational Databases are implicated by the satisfaction of the previous aspect. Instead of securing the datasets in any database, the system focuses on a secure transmission and outsources the problem to the area of responsibility of the data customers. The Real Time Security / Compliance Monitoring aspect is also largely eliminated by renouncing a longitudinal database. In total, however, even if the system bypasses these aspects, there are no weaknesses, so that we can consider the aspect of Data Storage & Distribution Concept as satisfied.

A-priori Performance and Feasibility: From an a-priori point of view, the system’s performance is limited by the user’s reaction time. Beginning with a potential customer’s request, we find, that an answer to such a request is bound to the vendor’s reaction time. This does not even grant the customer to receive an answer which may be a major problem if he wants to receive at least a certain number of datasets. The chosen, inelegant solution for secure data transmission is only a byproduct, which still requires a lot of time for encryption and decryption (as seen in Section 3.2.5), but which is insignificant when compared to any human reaction time. It is also highly questionable whether the defined system is sustainable. The intermediary instance has to control all traffic since the system does not provide any further automation. With a constantly growing amount of data, the effort for the intermediary instance is unlimited from which we must conclude that Future Reliability is not given.

Privacy: For a comprehensibly privacy evaluation, we consider what a malicious vendor, customer and a malicious central instance can do. Since we assume the data to be transmitted in an encrypted way, a listening instance can do, at most, as much damage as the central instance.

From a data vendor’s perspective, Unlinkability and Anonymity of their data are not given in case of the described attack. After submitting an answered survey to the mediating instance, this instance can link the dataset to the vendor’s identity. This already indicates that the aspect Non-Repudiation is hurt. Once the mediating instance receives information from him, there is no plausible possibility for a user to retrieve or deny this process. Disclosure of Information is also not given in a Status Quo approach-based system since once a user submits his data, he knows that he gives away further possibilities of influencing what happens with it. The user, however, must take this loss if he wants to use the advantages of the system (e.g. financial rewards) because he is not offered an alternative. The Non-Compliance aspect seems to be arguably because there is no possibility for a user to determine any compliances in association with his data. Nevertheless, since that concept is not in the sense of privacy we consider this aspect also to be unfulfilled.

From a data customer’s point of view, some of the previous determinations are transferable when it comes to the evaluation of a data customer’s privacy. Her data is Linkable, she is Identifiable and her data is exposed to the risk of Non-Repudiation. The Detectability aspect is also not satisfied because the intermediary instance knows about the surveys that a customer publishes. Disclosure of information, that is, the transfer of customer sensitive data to an instance without authorization, in contrast, is satisfied here because whenever a survey is transferred, it happens with the authorization
of the customer. Lastly, the Non-Compliance aspect is fulfilled because a data survey is only given to a customer who wants to answer it, while no additional data is published, so, each data transmission is made with the customer’s consent and in her interest.

5.3 PDS-based Approach

Section 3.1.1 and Section 3.1.2 already indicated, that the general concept of a PDS-based approach can be used in different manifestations for different use cases. This means, that we must define specifications for our intended implementation. Section 5.3.1 explains how we derive a solution for a Data Exchange Platform from the PDS-based approach, while based on that Section 5.3.2 analyses strength and weaknesses of the approach, based on our requirements and in an a-priori way.

5.3.1 PDS-based Approach: Sketch of an implementation

A solution for a Data Exchange Platform, based on the idea of Personal Data Stores can be sketched as follows: Each data Vendor owns a Personal Data Store with one interface for each customer. The PDS then is filled with the – unencrypted – vendor’s personal data. In each interface, the vendor can determine which datasets are accessible for her – and which are not. The customer can request the data then by communicating with her interface. An illustration of this architecture is provided in Figure 9. To ensure a secure authentication, we will use the authentication method that was presented in Section 3.2.1. Except that intuitive definition, we want to give a comprehensible overview of the functions that the different phases of a PDS-based approach offer.

**PDS-INIT:** The main functioning that can be matched to the Initialization phase is the creation of the Personal Data Stores and additionally one interface for each customer who is part of the system. Beyond this generation, there is nothing to do with these structure in the Initialization phase.

**PDS-GEN:** The Generation phase contains methods for the creation of data. Whenever a new dataset is created, it must be linked with a list of customers who can access it and the appropriate interfaces. Additionally, it is mandatory to create new interfaces in this phase, which may, for instance, be necessary, when a new customer joins the system.

**PDS-DECR:** The Decryption phase in this approach predominantly includes authentication and request methods. As mentioned above, we will implement a RSA-based approach for the authentication purposes, that should be usable in a comfortable way for users and customers. Since – compared to the ABE-based approach – the PDS-based approach does not prescribe any encryption method for secure data transmissions, we will implement an asymmetric encryption proceed to ensure anonymous and secure transmission of the data. For this purpose, we can take advantage of the fact, that vendors and customers know each other’s RSA public key since it is needed for the authentication process. Whenever a customer requests a dataset, it is encrypted with her public key and transmitted.

5.3.2 PDS-based Approach: A-priori Evaluation

**Functional Requirements:** As a considerable enhancement, compared to the Status Quo approach, users can enter their data to a closed system, not depending on any customer’s request. This allows
the users not only to “keep their hands” on the data initially, but it also is a fundamental basis for reusing data, after it is accessed the first time. To provide his data to multiple customers, the vendor only must make his data available to the appropriate customers. The system also fulfills the minimum functional requirements for a data customer, who can look through the vendor’s database to find the information she is interested in and request it, if she has the right to access it.

**Usability and Transparency:** The application does not completely fulfill the criterium of usability. The vendor must create and adjust a new interface for each new customer and additionally adjust each existing interface when he creates a new dataset. Additionally, the vendor must regularly check, if his customers still fulfill the requirements that are important for him. If a customers’ behavior no longer meets his expectations, he must adjust her interface and revoke her access rights. Through the possibility of re-using his data, the **Usability and Transparency** criterium is at least half fulfilled. The system is also highly transparent, which is a consequence of the architecture which – when carefully implemented – does not give an unauthorized party the possibility to access the user’s data.

**Useful Data Distribution Concept:** The architecture provides an interesting and completely distributed solution. This simplifies changes on the database level, providing a high degree in the aspects **Longitudinal Storage** and **Best Database Practices**. In addition, the solution finds a good method to face the challenges that come along with such a distribution. There is no problem with **Monitoring Compliance** aspects, which a vendor can completely control by adjusting his interfaces. Concluding we can say, that the Pos-based approach finds a good solution to supplement a completely distributed database so that the challenges are mastered in a good way.

**A-priori performance and feasibility:** The performance of this system is not depending on any human reaction time, which is a great improvement compared to the Status Quo scenario. The data customer can send a request and gets an immediate response. Depending on the decryption time of a dataset, the customer can read the answer within seconds. Nevertheless, the data can be adjusted individually at any time. The only drawback is that with a growing number of datasets or customers, each vendor must adjust his interfaces if he wants to continue using all the system’s capabilities.

**Privacy:** As an appropriate model for a privacy attack, we can assume a malicious customer, who tries to injure the vendor’s privacy by taking an advantage of the access rights that he has. Thereby, **Linkability** protection for data vendors is not given. Since an interface belongs to one data store at a time, the attacker can simply identify links between each data record that he can access from the interface he is logged into. This also concerns the **Identifiability** of a user, since it allows the data customer to identify the user within all the vendors he is related to. The system also leaves no margin to **Non-Repudiation**, since an attacker has an irrefutable knowledge base on the data that belongs to a vendor. Therefore, by using the interface concept, the **Disclosure of Information** criterium is fulfilled and ensured. At every time, a user can adjust whether a certain data record is accessible for a customer or not. Finally, the entire architecture also ensures, that these legislations are enforced consequently, which implies the **Non-Compliance** criterium.

From a customer’s point of view, an attacker would be a malicious vendor. This vendor has full knowledge and control over the customer’s interface in his **Personal Data Store** so that he can trace back each request and each transmitted dataset. Consequentially, whether the **Linkability**, **Identifiability** nor the **Non-Repudiation** challenge is defeated in this approach. To prevent **Detectability**, each interface could contain an overview of the accessible datasets. So, a customer could remain undetected when checking if she can access a dataset. The same applies for **Disclosure of Information**,
which cannot take place, without the customer’s permission or, in this case, even with the active action of logging in and purchasing certain datasets.

5.4 **ABE-based Approach**

In contrast to the PDS-based approach, the ABE-based approach is not derived from an entire, existing system design. Instead, it is an approach, built on the concept and taking advantage of the CP-ABE algorithm. Section 5.4.1 sketches the design of such a system. Based on that, Section 5.4.2 analyses strength and weaknesses of the approach, based on our requirements, in an a-priori way.

5.4.1 **ABE-based Approach: Sketch of Implementation**

The ABE-based approach is based on a centralized database – in contrast to the two former approaches. The datasets are encrypted with the CP-ABE scheme so that only customers who are authorized by their attributes can access a certain dataset.

Despite that intuitive definition, we want to give a comprehensible overview of the functioning that the different phases of a PDS-based approach offer. An overview of this system is provided in Figure 10.

**ABE-INIT:** The ABE-INIT phase consists of the generation of ABE-specific structures. The first step of the ABE-INIT phase consists of registering each existing customer in the Central Instance’ database. The Central Instance will later oversee the customers’ attributes, which is important, whenever a customer requests to flush her key. The central instance is also able to provide and invoke attributes.

**ABE-GEN:** The ABE-GEN phase mainly consists of data generation and encryption. In the ABE-based approach, each ciphertext is attached with a policy. Also, to make the data traceable, the user must publish which kind of data he has generated, so, for example, his favorite movie, his favorite holiday destination or his preferred party. This takes place in a database that contains only meta information on the actual database. During the ABE-GEN phase, the proceeds from ABE-INIT may also occur, that is, new attributes may be generated and provided or a customer flushes her private key.

**ABE-DECR:** In the ABE-DECR phase, a customer has the possibility to access to the datasets from the central database. After storing the encrypted data on a local machine, she begins to decrypt them with her private keys. Whenever her attributes qualify her to satisfy the policy, the decryption will be successful. Otherwise, the received dataset is useless for her.

5.4.2 **ABE-based Approach: A-Priori Evaluation**

A policy-based approach, as presented in Chapter 3.2, is a possibility to enhance the privacy aspects of the previous PDS-based approach, while adopting the fundamental benefits. The enforcement of access policies on the cryptographical level results in minimal contact between customer and vendor.

Functional Requirements: From a functional point of view, the system is not different from the previous ones. Vendors are still allowed to create data and determine access restrictions as far as desired. Furthermore, vendors can create new datasets at every point of time, even if currently no
customer is interested in the data. This also applies to customers, who can collect certain, similar datasets by browsing through the database, independent of the vendor’s current activity.

**Usability and Transparency:** In contrast to the previous approaches, the usability requirement is completely met in this approach. When creating data, the vendor can protect it uniquely with a policy, which is valid and sufficient for all data customers. New customers and new datasets do not exert any influence on any existing dataset or policy.

**Useful Data Distribution Concept:** The architecture provides a compromise between central and distributed data. The data vendor can create and protect his records in a protected environment. He can then rely on the cryptographical access enforcement, which the $Cp$-ABE algorithm implements. This allows him to send his records to a centralized, public database and makes sure that only permitted customers can access his data. Applicable solutions for the database are manifold, one example is a blockchain which provides transition monitoring and security aspects but is still very feasible for Big Data oriented databases. Bearing this in mind, as at least one possible solution, allows us to consider the aspects of Longitudinal Storage and Best Database Practices as fulfilled. Compliance Monitoring is enforced on the cryptographical level, while Secure Computations take place in distributed local environments. As a concept, this seems quite convincing, nevertheless, we will still have to look at the approaches feasibility.

**A-priori performance and feasibility:** The performance of this approach is bound by the limits of the attribute-based encryption scheme. We have seen that $Cp$-ABE scheme can be significantly slower than those that are applied in public-key cryptographic systems. Therefore, $Cp$-ABE provides functionality on the cryptographic level. From our current point of view, we cannot say if we can take enough advantage of this functionality to compensate the given performance disadvantages.

**Privacy:** The $Cp$-ABE approach covers our privacy aspects advantageously. Especially by storing the data in a centralized structure, anonymous access is better supported than in previous approaches. As an attacker model, we may now assume a malicious vendor or customer, or an attacker who gains access to the central storage environment.

This concept solves the problems of Linkability and Identifiability for customers. Neither a customer nor anyone else with access to the platform can draw any conclusions from the mere existence of the data as to their origin, due to the unknown origin of the data. So, the Non-Repudiation challenge is solved in the ABE-based approach. Thinking one step ahead, it also prevents an attacker to state any provable claim. As discussed earlier, the Disclosure of Information, that is a reliable enforcement of determined policies, is assured on the cryptographical level of $Cp$-ABE. Lastly, the Non-Compliance aspect, that is the non-following of data protection legislation is completely solved and – as mentioned – enforced on the approaches cryptographical level.

The data customer can also benefit from the ABE-based approach. Having an intermediary platform also enables him to anonymously access vendor’s data. Therefore, logging the access operations – or attributing two different access operations to the same customer – is not possible, so that the requirements that relate to Linkability and Identifiability are completely satisfied. This also implies Non-Repudiation for customers. Furthermore, there is no convincing argument which an attacker could use to undeniably demonstrate that a customer accessed a certain dataset. This is related to the Detectability challenge, which is faced well. Finally, this approach is our first approach, where a customer may not give due notice, that he wants to access any data, so that information about his requests is not passed on, resulting in fulfillment of the aspects Disclosure of Information and Non-
Compliance. Summed up, when compared to our requirements, this approach fulfills each of our privacy requirements.

5.5 Recapture and Comparison

In Table 5 we can find a schematically summary of the results from the previous sections. From a decision theoretical point of view, as for example contributed by van Nitzsch [van Nitzsch 2015], we can conclude that the PDS-based approach entirely dominates the Status Quo scenario, based on our requirements, which means, that the PDS-based approach is at least as good as the Status Quo scenario in each criterion. For this reason, we can completely exclude the Status Quo scenario from our further considerations. Since we only have a vague statement on the ABE-based approaches Performance and Feasibility – when applied in an entire system – we will follow the two remaining approaches. To compare the last criterium, we will implement both approaches in a comparable way so that we afterward can find out if there is a gap in our last remaining criterium and what this gap means for the aim of this thesis.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Data Vendor</th>
<th>Data Customer</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status Quo</td>
<td>✗✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>PDS-based</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Abe-based</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

6 Realization

This chapter provides an overview of the software design and the implementation. The implementation will be carried out in Eclipse IDE for Java EE Developers from the Eclipse Kepler SR2 Packages. The chosen version offers some advantages that are mandatory for a smooth implementation. By taking advantage from the functionalities of the development environment (DE) it is just a small effort to run the chosen ABE-library, especially compared to experiences of other people with similar libraries.

6.1 Software Design

To make both approaches comparable, functional and cryptographical aspects are implemented completely in Java language. Especially for the secure authentication and for the cryptographical pairing processes that the approaches are based on, there may be faster solutions achievable by using a low-level programming language and a Java wrapper. However, this would make the comparability of both approaches doubtful and would also break the isolation of the Java-based system, which would, for example, complicate a porting to an Android- or IOS-based device. The general advice for an application that provides a big number of unstructured data would be the use of a Not Only SQL (NoSQL) database, as Rajan et al. advised [Rajan et al. 2013]. Since we will not simulate the application as a distributed network of vendors and customers, but only want to evaluate multiple access processes, we will forgo a designed database. The main reason here is, that the additional implementation effort would only lead to access processes of both approaches which would cancel each other out if implemented correctly. We will use default Java structures instead, namely Lists and Maps to store the data.

As it is very common in software development, we will present the high-level system overview as a multi-tier model. The three-tier architecture, consisting of the presentation tier, the logic tier and the data tier is the most common one in software architecture. A contribution of this model is done by Ramirez [Ramirez 2000], but we will only use the model for a general illustration of our implementation. Our architecture sketch is illustrated in Figure 11. At the lowest layer, we find Java and Java libraries, which is not included in any of the three layers and just emphasizes, that there is something behind the data tier, that compiles and processes our application. The Data Tier consists of the Data Management Layer. This layer includes classes that control the data and data flows, but also the data bases for both approaches which are additionally implemented as classes.

The next superordinate tier, the Logic Tier includes the cryptographic primitives for the approaches. The concurrent layer includes all possible configurations and is important for each superordinate class. An intuitive use case for this is, for example, the adjustment of the applications security level.

The last tier, which is the Application Tier, then includes the application groups, which are functioning combinations of cryptographic primitives and data and, lastly, the Application Layer provides demonstrations of the application, where the different parties can interact. When describing the
design in the consecutive approaches, we will follow the diagram in Figure 11 in a top-down manner beginning with the application layer and ending with the data management layers and their relation to the included Java libraries.

6.1.1 Design of the Application Layer

For the Application Layer, we aim to implement various testing environments. Testing functioning should be distributed in useful divisions, for example, tests in the context of ABE, tests in the context of PDS, tests for generation of data exchange specific structures. Furthermore, tests should be classified, so that especially long-running tests can be excluded by setting a threshold parameter that is in charge of this matter. Except for these atomic tests, the other classes aiming testing purposes are implemented to picture a running instance of the system in its entire perimeter. For a fast and dynamic testing, a simple application is implemented that allows a dynamic generation and interaction between instances of classes. For this purpose, an application with a parser which is connected to the command line is well-suited due to its low implementation effort and a high flexibility in testing. The parser should implement each available, for this purpose appropriate, function in a useful way. Additional information on the parser is provided in Appendix B. Lastly, and based on the parsers concept, a static testing environment is implemented. This testing environment offers a certain number of variables that can be set initially to determine the limits of the system. A system that is bound to this limit then is set up and a use case that represents the behavior of the vendors and customers is carried out. For each run of the program, certain parameters can be set as invariants, so that the system is set up multiple times with different shapes of the invariants. While the use case runs, time measures are captured automatically and stored in an environment that allows a more specific analysis afterward.

6.1.2 Design of the Application Group Layer

The application group layer is a layer that is very close to the application layer but also contains logical elements. The application groups in specific are vendors and customers, as we have entitled and defined them in previous sections. The classes which go with them bundle multiple methods from underlying layers to provide functioning that represents the aimed functioning for both application groups. For example, the application groups bundle functions of the encryption and the data model layer to store data in an encrypted way. Furthermore, they offer methods to communicate with each other, using unique identifications (IDs). The application groups are relevant when it comes to implementation, but for software design purpose, we only want to keep in mind, that they implement multiple methods that are necessary to make easy testing possible later.

6.1.3 Design of the Cryptographical Layer

Requirements for the cryptographical layer can be derived from the previous section in a straightforward way. The layer itself is encapsulated and consists of classes that implement the primitives that are relevant for ABE and RSA. Vendors and customers can use these classes for encryption purposes once they are initialized.

6.1.4 Design of the Configuration Layer

The configuration layer includes parameters and settings but also predefined errors that are useful for testing and debugging purposes. The configuration parameters are defined as public static final, which indicates, that they are globally valid, that they are not in conflict with each other and that they cannot be changed at runtime.
6.1.5 Design of the Data Management Layer and Other Layers

This section presents the last layer that we must implement actively and the last tier of our tier-based software architecture. The Data Management Layer is the layer where all the encrypted data arrives and must be stored in an efficient and logical way so that both approaches - the ABE- and the PDS-based approaches – can run concurrently and independent of each other.

Generally spoken, this layer is where the two approaches diverge the most. While the PDS-based data is stored in a distributed way, the ABE-based data is stored in a big, unstructured database. We have discussed properties and advantages of both system, but we want to recapture shortly, that the PDS-based approach is based on a processing unit that complements the personal data store and differs between access entitled and unentitled customers, while the ABE-approach manages this on encryption level, so that the encrypted data is attached to an access policy. Despite this difference, both approaches should use a minimized but similar data model that simplifies the process of finding a certain information for customers. While for the PDS-approach this only means, that a customer must send the data ID of the information that she desires, for the ABE-approach that means, that there must any kind of plaintext wrapper for each dataset, which describes at least what the dataset contains. Our contribution will use two databases for this purpose: One database (DB1) contains the pure data as ciphertext, another database (DB2) contains plaintext information, to be more specific: The data ID, so the kind of information that is contained and the address of the ciphertext. It is imaginable to also publish the access policy in this database so that a customer can skip the search process if his attributes do not satisfy the access policy anyways.

The main reason for this concept is its extend-ability for practical usage. If DB1 is located on a public server, a malicious user may try to distort or delete datasets. DB1 could be designed as a blockchain for this purpose. Each change of a dataset would be noticed so that the encrypted data is resistant against malicious users. DB2 would then provide the locations of the data in the blockchain. Additionally, payments could be transacted on DB2, for example, a customer could look up the data ID and the access policy for free, then transact an amount of money and receive the address of the data in DB1. We will further discuss this in Chapter 8.

6.2 Implementation

Section 6.1 provided a conceptual overview of the general design of the two approaches realization. This section now deals with the implementation of both approaches. Determinations from the previous sections are used as a basic for the implementation. This includes, for example, the logical division into the predefined Initialization, Generation/Encryption and Decryption phases.

Section 6.2.1 reuses the implementation sketch that was presented in Section 6 – but supplements the Java classes that will be implemented. This section also explains which libraries are used for the implementations. Section 6.2.2 then presents our approach to generate datasets for our Data Exchange Platform. Based on Figure 11 and Figure 12, Sections 6.2.4 and 6.2.5 each sketch the implementation of one of the two approaches – The ABE-based approach and the PDS-based approach.
6.2.1 Detailed System Overview and Libraries

Figure 12 Extended Three-layer model

Chapter 6 already provided a coarse-grained overview of the system, as a tier-oriented diagram. Figure 12 now picks up that information and supplements it with the actual implemented Java classes.

6.2.2 Data Generation and Data Model

For our approaches, the content of the data is not relevant. Dimensions that we want to influence are the number and the size of data sets. We generally consider data as an array of characters (String). A character is represented by one byte (8 bit) of information in most encoding standards. Java stores characters without redundancy, so that we can assume 1000 characters to be 1 kilobyte (kB). We will use this property to generate data of arbitrary size. The parameter for the dataset size is introduced in Section 6.2.6.

The data model is a very lean class with little functionality, but is used to give both approaches a common shape, by defining and providing the meaning of a dataset, that is – for example – the kind of data that is stored in a dataset (favorite movie, favorite holiday destination, salary etc.). For the Abe-based and the Pds-based approach, we will find an implementation where a customer can see a dataset’s meaning if he wants (which we called browsing datasets before). However, the permission to read the content of a dataset is independent of this.

6.2.3 Libraries

For the Abe-based approach, we can choose between multiple implementations. Zickau et al. [Zickau et al. 2016] present 14 different libraries, implementing an Abe approach. Three of these libraries implement the Bethencourt et al. Cp-ABE scheme in Java language: cpabe, JCPABE and DET-ABE. cpabe is a predecessor of JCPABE, where JCPABE improves the cpabe approach. DET-ABE has a slightly different functionality and encrypts all data with the AES symmetric encryption algorithm and secures the AES keys with the Cp-ABE algorithm. This does not correspond to the principle we want to evaluate, so the library is not interesting for us. For this reason, JCPABE is the best library for us. JCPABE implements the cryptographic basics as defined in Section 3.2.4. In addition, the library manages attributes by name and offers an automatic policy parser without policy optimization.
For the PoS-based approach, we will need to implement a secure authentication method, which was already presented in Section 3.2.1. The RSA algorithm is based on the standard Java libraries `java.security.*` and `java.crypto.*`. `java.security.*` provides manifold methods to generate and store keys as Java-objects, choses the cryptographical algorithms and generate random values securely. The second library provides a secure padding algorithm, to enable the encryption of plaintexts that are shorter than the block-size of RSA.

6.2.4 The ABE-based approach

For the ABE-based approach a customer receives her attributes initially by a central instance. Whenever she receives new attributes she can refresh (flush) her private key $SK$ that she needs for the decryption process. Therefore, she sends a request to the central instance. The central instance then uses the KeyGen algorithm (cp. Section 3.2.4) and delegates the generated key to the customer. For a vendor, it is important to encrypt his datasets under a certain policy. Therefore, he needs to know each attribute that a customer can have and its meaning. The encrypted datasets are stored in an unordered list which is an unordered structure of encrypted dataset. To find a desired dataset the customer can find out the addresses of this data in a meta list and poll it from the main list afterward.

AbeCentralInstance.java is implemented as proposed in the Bethencourt et al. [Bethencourt et al. 2007] approach. It supplies attributes to the customers, when the system is set up or dynamically during run-time. Further functionality of the AbeCentralInstance class is managing the attributes and flushing the customers keys after they obtain new attributes. When flushing a key, AbeCentralInstance uses its internal assignment table to track who of the customers has which attributes.

The important structures for the ABE-based approach that are located on the lowest tier — the Data Tier — are the DataChain and the DataChainMeta, which both can be seen as unstructured collections of elements. The DataChain contains elements of the kind DataChainSet consists of an encrypted message and a random address that represents the position in the data chain. Two arbitrary, disjoint datasets may not have the same addresses. The DataChainMeta is utilized to find elements in the DataChain. Each element in the DataChainMeta, namely each DataChainMetaDoc contains information on one related DataChainSet that it is related to. It is also conceivable that a customer can browse the DataChainMetaDoc unrestrictedly except the address field. Whenever she finds an interesting dataset, she can purchase the address for a minimal fee. For our performance evaluation, the division of the two chains is not necessary.

---

AbeCentralInstance.java

```java
// List of Attributes
+ Map<Integer, AbeAttributes> universe;
// Customer-Attribute-Matrix
- Map<Integer, List<AbeAttribute>> cusAttMatrix;
// Keys
- AbeSecretMasterKey abeMasterKey;
+AbePublicKey abePublicKey;

// Attribute & Key Generation
- void createAttribute(String name, String description);
- void provideAttribute(int customerId, int attributeId, boolean keyFlush);
+ void flushKeyABE(int customer);
...```

**Figure 13** Overview of Central Instance managing the ABE approach
An exemplary schedule for the Abe-based approach can be found in Figure 15. In this schedule, the AbeCentralInstance initializes the processes by generating $PK$ and $MK$ and publishing Key $PK$. The AbeCentralInstance then tells the customers their attributes. Thereupon, the customer
requests a secret key $SK$ that represents her attributes. After a successful authentication, the AbeCentralInstance generates and sends this secret key to the customer. In our exemplary schedule, next, vendor 1 uses the public key to create a dataset (Data 1 in Figure 15) and encrypts it under a policy (ATTRIBUTE1 AND ATTRIBUTE2). Customer 1 locates the dataset (using the DataChainMeta) and downloads it. By using her secret key $SK$, she can decrypt the ciphertext, thus access the dataset. The same does not work for Customer 2 in our schedule because she lacks ATTRIBUTE2.

6.2.5 The PDS-based approach

In the PDS-based approach, vendors do not only have to protect their data, but also manage it, including storing them and managing the access rights. Each new dataset needs a list of customers as an attachment, which determines who is allowed to access it. Arbitrary management of this list, that is providing or revoking access for customers, must be possible at every time. Whenever a customer requests a dataset he must authenticate, using a public token, as explained in Section 3.2.1. When an authentication process is successful and the customer is permitted to access, his request will be

<table>
<thead>
<tr>
<th>Vendor.java</th>
</tr>
</thead>
<tbody>
<tr>
<td>//The (individual) Personal Data Store</td>
</tr>
<tr>
<td>- PersonalDataStore persDS;</td>
</tr>
<tr>
<td>//The (individual) Map of all customers and their interfaces</td>
</tr>
<tr>
<td>- Map&lt;Integer, AbeAttribute&gt; cusIntMatrix;</td>
</tr>
<tr>
<td>//InitializationMethods</td>
</tr>
<tr>
<td>- createNewInterface(int cusId, List&lt;Integer&gt; approvedData);</td>
</tr>
<tr>
<td>- createData(int dataId, String value, List&lt;Integer&gt; approvedCustomers);</td>
</tr>
<tr>
<td>//Management Methods</td>
</tr>
<tr>
<td>- approveAccess(int customerId, int dataId);</td>
</tr>
<tr>
<td>- RevokeAccess(int customerId, int dataId);</td>
</tr>
</tbody>
</table>

Figure 16 Illustration of the Vendor class

successful. On generation, each vendor constructs a PersonalDataStore and creates his initial datasets. The datasets are attached with additional information, that is the data domain (as ID), the value (the entry) and a list of access approved customers. For each of these customers, a PersonalDataStoreInterface is generated an adjusted, so that the corresponding customer can access the datasets for which she has the permission. Additionally, a method for directly creating a PersonalDataStoreInterface is provided, where a determined customer receives access to certain datasets. Lastly, access permissions can also be managed individually, by approving or revoking a single access permission at an interface. All methods are secured against multiple calls so that, for example, a data customer cannot have two corresponding interfaces in a single PersonalDataStore at any time. The customer, on the other hand, only needs to perform an authentication process to access her data. As soon as she is connected to her interface, she can request a data Domain ID and receives the corresponding dataset – if it exists – and she is authorized to access it.

Figure 17 shows a possible schedule of an PDS-based system with two customers and one vendor. The Vendor creates one dataset and two interfaces for each customer. He permits access for one customer.
Afterwards, he generates one dataset that he stores in his Personal Data Store (Data 1). He allows Customer 1 to access it. Not-allowing permission to any Data (as with Data 1 for Customer 2) means the same, as denying this permission. If Customer 1 now sends an authenticated request to his interface, the interface will determine that Customer 1 has the right to access Data 1 and send it back to him. If Customer 2 tries the same, his request is denied.

6.2.6 Implementation of the Evaluation Application

Based on both approaches the evaluation suite is designed and implemented. The evaluation is implemented in Evaluation.java and can best be executed from the EvaluationWrapper.java class. The EvaluationWrapper automatically executes an evaluation multiple times and stores the results in csv file format so that the results are available in a structured form.

The Evaluation class itself automatically generates a customizable number of vendors and customers. For each vendor, a customizable number of datasets are generated. The size of all these datasets are equal and also customizable. Specific for the A8E-based approach, we can also determine the depth and height of the policies and the number of attributes. An interesting parameter which we log during the evaluation phase is the number of successful access processes which can, for example, be represented by a percentage value of the number of access processes divided by the number of successful access processes.

Figure 17 Proceed of a Data Exchange in a PDS-based System
7 Evaluation

This section describes the evaluation of the two implementations. Section 7.1 describes the evaluation of the ABE-based approach which is a less intuitive since it is influenced by more parameters than the PDS-based approach. Section 7.2 describes the results of the comparative evaluation. Section 7.3 describes what the results mean for us and in section 7.4 our results are recaptured and summarized.

7.1 Evaluation of the ABE-based approach

For the ABE-based approach in specific, we will evaluate the impact that the number of attributes, the policy length (tree-width) and the policy height (tree-height) has. The results of this evaluations can partly be compared to former literature, as Ambrosin et al. [Ambrosin et al. 2015] who considered the effect that the number attributes have and Bethencourt et al. [Bethencourt et al. 2007] who made firstly practical, but especially theoretical considerations that affect the complexity of the policy. But there is no former evaluation that describes the performance of the approach in a Data Exchange Platform similar application. In each test, the performance time is measured in milliseconds (ms) which can be found on the y-axis, while the x-axis represents the evaluated value.

7.1.1 Impact of the Number of Attributes

We will begin with the comparison of the impact that the number of attributes has. Our testing environment is set up with 30 customers and one vendor. The vendor owns one dataset with a size of 100 bytes and each customer will receive all attributes so that each customer is permitted to decrypt this dataset. The evaluation now describes the impact that the number of attributes has on the performance. The evaluation was carried out twice, for a policy length of 25 and for a policy length of 50.

Figure 18 Evaluation of ABE-based approach, Number of Attributes (Initialization, Generation, Decryption, Total)
7.1.2 Impact of the Policy Length

Our policy length tests include 30 customers and one vendor. The vendor owns one dataset with a size of 100 bytes and each customer will receive all attributes so that each customer is permitted to decrypt this dataset. The evaluation now describes the impact that the policy length has on the performance. The evaluation was carried out twice, for 5 attributes and for 50 attributes. The results are divided into the three defined, main-phases (Initialization, Generation and Decryption). The last diagram represents the sum of the three phases (Total).

7.1.3 Impact of the Policy Height

Our policy height tests include 30 Customers and one Vendor. The vendor owns one dataset with a size of 100 bytes and each customer will receive all attribute so that each customer is permitted to decrypt this dataset. The evaluation was carried out twice, once for 25 attributes and once for 50 attributes. The policy height is implemented as a path in the access tree so that a policy height of 5 may, for example, be represented by the policy $ATT1 \text{ AND } (ATT2 \text{ AND } (ATT3 \text{ AND } (ATT4 \text{ AND } ATT5)))$ where the universe consists of exactly the 5 occurring attributes.
7.1.4 Plausibility of the Results

The results that the evaluation of the number of attributes brought, can be compared to the contribution of Ambrosin et al. In our evaluation, generation and encryption time are nearly constant, when the number of attributes increases. When we compare this with Ambrosin et al.’s results, we see that they achieve similar results on a laptop device, where the duration only increases very slightly. However, the gap that we can observe between those two phases and the key generation phase, that takes a relevant longer time in our contribution, is not congruently to Ambrosin et al.’s results. We can find a justification for this difference in two aspects: On the one hand, Bethencourt et al. verify a linear growing complexity in the key generation phase, recapturing, that “The key generation algorithm requires two exponentiations for every attribute given to the user.” [Bethencourt et al. 2007, S. 326] On the other hand, we have to supplement, that the initialization phase, as defined in Section 5.3.1 is not equal to Ambrosin et al.’s Keygen phase. Initialization, for example, also includes the data and the attribute generation.

The results from the policy width and policy height reproduce assumptions that Bethencourt et al. made. Changes in the policy do not influence the performance of the initialization phase, which only depends on the number of attributes. The generation phase only depends on the number of nodes in the access tree, which grows linear when the policy grows in its width or height, which it exactly does here. The decryption process needs “one exponentiation for each node along a path from [...] a leaf to the root”, which exactly indicates, why the decryption process becomes slower when the policy height grows, why the policy depth has a much smaller impact on the performance.

7.2 Comparison of both approaches

While section 7.1 dealt with the ABE-based approach alone, this section considers the performance of both approaches, when they run concurrently to fulfill an identical use case. Section 7.2.1 presents the difference that is caused by the size of the accessed dataset. Section 7.2.2 presents an evaluation on the detectability of non-permitted datasets. Section 7.2.3 lastly evaluates the feasibility of both systems when access processes are applied multiple times.

7.2.1 Impact of the Data Size

First, we evaluate the performance of both approaches, based on the size of the transferred data. Our test includes one vendor and 30 customers and each customer will receive all attributes so that each customer is permitted to decrypt this dataset. The evaluation now describes the impact that the data size has on the performance. The evaluation was carried out twice, for both approaches. As the performance of both approaches is already far apart at 100 kilobits, we can stop the evaluation here.

Figure 21 ABE-based approach, Evaluation of Policy Height (Initialization, Generation)
We can assume that the performance of the ABE-based approach does not drop under the one of the PDS-based approach after 100 kilobits. The reason for this is that the PDS-based approach clearly indicates a linear trend, while the block size of the ABE-based approach at 100 kilobits has not yet been reached. Even if this block size is reached, however, the maximum time required for the process can be assumed to mostly be doubled, which at 100 kilobits is even greater than the PDS-based approach. The reason for this assumption is that if the performance is worse than double, we would just split up the dataset in two parts initially and then use the appropriate proceed twice.

### 7.2.2 Impact of Non-Permission Detectability

Our non-permission detectability test includes 50 customers and one vendor. The vendor owns 100 datasets with a size of 35 Kbit. The customers are not permitted to access each of these datasets in this evaluation, but the number of accessible datasets varies. The x-axis presents the share of the 100 datasets that the customers are allowed to access. This value varies from 4 to 92 accessible datasets. The result is presented, reduced to the total time in Figure 23. For the ABE-based approach, furthermore, we use 15 attributes per customer. At this point, we would also like to remark that the data size is chosen in a way so that both approaches are equally fast at the highest value. As section 7.2.1 indicates, a smaller data size would shift the point of intersection to the left, a bigger size would drive the graphs apart.

---

**Figure 22** Comparison of Data Size (INIT+GEN, DECR, OVERALL)

**Figure 23** Comparison of Non-Permission Detectability
7.2.3 Impact of Scaling System

Our last test evaluates the impact of a varying number of customers and vendors. Each vendor has one dataset of 25 Kbit size. The customers will receive all attributes in the ABE-based approach and all access rights in the PDS-based approach. That leads to a total number of access processes of $|\text{Number of Customers}| \times |\text{Number of Vendors}|$. The evaluation now describes the impact that different compositions of customers and vendors have. The x-axis represents the product which describes the total number of access processes. The evaluation results are represented as single data points. Due to the different composition, in some cases, these data points differ from each other when considering the same x-axis value. Nevertheless, a compensating straight line, as shown in Figure 24, indicates a clear trend for both approaches. As we would expect, due to the generation of multiple encryption-relevant structures, the ABE-based approach has an initial overhead. As a result, even with a minimum of access processes, the performance is around 60000ms, while the PDS-based approach starts in the origin. Nevertheless, at around 260 access processes, both approaches indicate the same performance.

7.3 Meaning of the Results

The comparison of the two approaches brought us three important insights. Firstly, we have seen that the ABE-approach we used is capable of sending a large amount of data efficiently. Due to the chosen, anonymity granting, asymmetric encryption in the PDS-based approach, this approach shows a linear performance drop, when increasing the dataset size. As a result, the ABE-based approach is better in handling large datasets than the PDS-based approach is. Our second comparison evaluated the detectability of non-permitted datasets. The ABE-based approach here offers the a-priori advantage that a customer can check if he is able to satisfy a policy before he tries to decrypt the appropriate ciphertext. In order to find out the accessibility in the PDS-based approach, an authentication at an interface is always necessary. We have seen, that this difference has a strong impact on the performance, with which the ABE-based approach reveals another advantage compared to the PDS-based approach. The last evaluation showed, that the ABE-based approach can compensate its
initialization after few access processes. It is important here, that the ABE-based approaches performance is significantly better than the performance of the PDS-based approach, but it is also remarkable that the point of compensation is reached after very few access processes.

7.4 Recapture and Overview

The evaluations that we considered had two main objectives: The evaluations in Section 7.2 dealt with the ABE-based approach and contributed an idea as to its practical scaling. We have seen, that the number of attributes, the policy height and the policy length all just have a linear impact on the ABE-based approaches performance. Section 7.3 then dealt with a comparison of both approaches. This comparison focused on the performance when varying the data size, considering the non-permission detectability and lastly considering both approaches in a natural system with arbitrary many access processes. All three evaluations indicated an advantage for the ABE-based approach.
8 Discussion

We have seen that the ABE-based and PoS-based approaches provide a user- and privacy-oriented solution for a Data Exchange Platform as requested in our thesis goal. The ABE-based approach in its ciphertext policy manifestation offers a sufficient solution for an enhanced Data Exchange Platform. We can easily create a system with several hundred attributes and long policies. The good results in general performance questions with the ABE-based approach show that neither the initial overhead nor a generally poor performance hinders the feasibility of this approach. The feasibility of a large attribute universe also allows us to implement negative attributes as follows: Attributes are ordered to a domain. Each customer has either the positive attribute “Att” or the negative attribute “Not:Att” from this domain. Since we have shown that a large number of attributes scales well, this scheme does not cause any problems when applied practically. Additionally, we have also seen that the policy height has a strong impact on the performance so that policies with a too deep nesting should be avoided whenever possible. So, for practical systems, policies should be optimized where possible. We leave the problem of constructing an optimal policy open here, but want to remark, that the minimization of non-negative Boolean formulas (policies) is in coNP [Bloniarz et al. 1984], where we generally assume coNP = NP ≠ P, so that the minimization and optimization problem for policies may not be solvable efficiently at all.

The reason why the PoS-based approach did not perform efficiently was that due to the fact that, apart from a number of unfulfilled privacy aspects, the performance could not outperform the ABE-based approach. The main disadvantage for performance properties was the use of an asymmetric cryptosystem. Asymmetric cryptosystems have a minimal initial overhead because after the generation of the private and public key it is only necessary to broadcast the public key to each customer. Therefore, the asymmetric cryptosystem has an unsatisfactory performance when it comes to encryption and decryption processes itself. This performance situation would differ when using a symmetric cryptosystem. In a symmetric cryptosystem, first, an asymmetric cryptosystem is used to exchange mandatory keys. Then a secure and fast channel between both communication partners is set up. The exchange of messages in this channel then is much faster, compared to the asymmetric approach. Therefore, such a symmetric cryptosystem has two main disadvantages when compared to our use case. First, in our use case of a Data Exchange Platform, we assume the datasets to be very small. The overhead of building a symmetric communication channel is simply too large to pay off. The safe construction of the system alone needs an exchange of some bytes and these bytes could already be used for transferring an entire (small) dataset before the symmetric channel is build up. This argument may no longer apply when we consider for example an exchange platform for multimedia data. Then, however, another disadvantage of the symmetric cryptosystem would become relevant. Whenever a party wants to communicate with its interface, the other party can see what actions the first party takes. This is a required by the symmetric channel. Then, for example, when a customer wants to know if he is allowed to access a certain dataset, the vendor will know. This affects the LINDDUN requirement Detectability, because the vendor then knows that the customer is interested in a certain dataset.

For both approaches, we have not clearly discussed an implementation of payment methods until now. For the PoS-based approach, this process can be realized for example on the interface. Whenever a customer requests a dataset and has the permission to access it, he has to pay a little fee before the interface forwards the dataset to him. For the ABE-based approach, we proposed using two chains for
this purpose. The first chain is unordered and contains each dataset that is generated by a data vendor. The second chain contains one element for each element in the first chain. Here, each element contains meta data on the corresponding element from the first chain. Additionally, there is one link to the corresponding element in the first chain. A conceivable solution may be now, that a vendor can purchase the address of an element in the second chain. So, he can find out information on the dataset, before purchasing it, but when he wants to read the vendor specific entry, he has to pay for this information. In practice, the first chain could, for example, be realized as a blockchain. That is, changes in the first chain, as forgeries by a malicious third-party, could be traced back. Principles and applications of the blockchain technology were for example contributed by Olleros and Zheng [Olleros and Zheng 2015] with assistance of Pilkington. We leave the implementation of a working blockchain for the ABE-based approach open. Another, practical, problem in context with this approach is the useless purchase from a customer's point of view. When a customer buys an element in the first chain, that does not mean that he automatically is permitted to decrypt it. One possibility to solve this problem would be, to attach the access policy to the elements in the second chain. This would warn a customer not to purchase datasets that are not accessible for them, but it would not stop her completely.

For further research purposes, the comparison with the traditional private-public-key encryption-based approach showed that use cases were the ABE-based approach is much more suitable exist. From the author's point of view, the results in this work indicate that research should focus more on the practical implementation of ABE-based systems. Many research efforts are currently focused on improving individual phases of the ABE process, like for example outsourcing schemes, as outsourcing the decryption process [Green et al. 2011] or introduction of decentralized [Lewko and Waters 2011] or mediating [Ibraimi et al. 2009] systems for dissolving authorities. All these approaches have been scarcely or not evaluated, so there is currently confusion about the feasibility of attribute-based systems in general. The CSO magazine\(^{14}\) even warns that the implementing security companies ignore the ABE-based approaches because of the unclear performance. From the author's point of view, science is particularly in demand to construct and evaluate solutions for larger systems such as social networks or document clouds to test the feasibility of such solutions and appropriately indicate potentials, so that ABE-based systems will ensure the security and privacy of our data in the future.

\(^{14}\) https://www.csoonline.com/article/3211440/security/whats-the-roi-on-attribute-based-access-control.html
9 Overview

The goal of this thesis was to provide an application that can be used as a framework for an online Data Exchange Platform with enhanced economical and privacy aspects, which may be controlled by but is not accessible to a third-party data broker. The problem of the Status Quo approach is, according to the extended privacy model LINDDUN, that important privacy aspects like Unlinkability and Anonymity, are neglected. Another remarkable disadvantage that we have identified is that vendors cannot exclude undesirable customers from accessing their data. Lastly, the Status Quo approach is completely based on an intermediary instance that can monitor the entire data flow. In order to find a more suitable solution, we have looked at approaches that are generally used as so-called Privacy Enhancement Tools (PETs). These describe multiple attempts to analyze privacy and security threats – for example, the LINDDUN privacy threat model – or tools that provide enhanced privacy for certain applications – as for example the approach of Personal Data Stores. Because for our use case of a Data Exchange Platform we have to assume very sensitive user data, we also considered secure data transmission schemes. For this purpose, we considered multiple cryptographical schemes, like the Group key-based and the Attribute-based Encryption scheme. Furthermore, we introduced RSA as traditional private-public-key-based asymmetric encryption scheme and a method for secure authentication based on RSA.

In order to find a solution for our use case from these approaches, we looked at various challenges that modern data architectures, for example in the Big Data context, must face. With the help of these challenges, we have identified the main requirements for our Data Exchange Platform. We determined that it is important for the Data Exchange Platform to ensure basic functionality and usability. Furthermore, the storage concept and performance were of great importance to us. In addition to these criteria, we also identified security and privacy aspects as important challenges for our Data Exchange Platform.

Next, we looked at the initial approaches and PETs with the help of the requirements. Approaches that are already common today, like Group key-based encryption are a first promising idea but still not feasible enough to realize a system with such a big amount of data as we are aiming to do. The more novel attribute-based encryption approach, therefore, surpasses the Group key-based encryption schemes from a functional point of view. Additionally, the approach of Personal Data Stores for enhanced privacy proved to be a suitable solution for our use case. For the ABE-based approach we, proposed a completely centralized, public database, where all datasets are stored. The data vendors access conditions then are enforced on the cryptographical level. For the PDS-based approach, the data is distributed and stored on the data vendor’s side. Interfaces in the Personal Data Stores manage and enforce the access conditions.

We found that both approaches entirely dominate the Status Quo scenario, from a decision theoretical point of view. That is, for each requirement, the two approaches at least as good as the Status Quo scenario or even better. The PDS-based approach offers a reliable performance and a solid fulfillment of the privacy requirements. However, we found that the PDS-based approach does not entirely satisfy our privacy requirements. Due to the distributed approach, vendors and customers still are in a close relationship and extended privacy threats, like Linkability and Identifiability are not eliminated. The ABE-based approach, on the other hand, completely fulfilled our privacy requirements. Therefore, we were not yet to determine further performance aspects on an a-priori way. Because based on this
information, we have not yet been able to determine any of the two approaches as better, we decided
to implement both approaches.

We contributed an evaluation framework for both approaches where for an arbitrary number of
vendors, customers, datasets and an arbitrary dataset size, measurements can be carried out. For the
ABE-based approach, additional evaluations were possible, as for example the impact of the number
of attributes, policy lengths or policy heights. The comparison of the two approaches brought us the
needed information on the performance. We began to evaluate the ABE-based approach alone. We
found out here that the ABE-based approach allows for an arbitrarily scalable system. Policy length,
height and the number of attributes only have a linear impact on the overall time performance. This
speaks to the fact that the ABE-based approach is suitable to build a privacy oriented Data Exchange
Platform. Additionally, we evaluated both approaches in a comparative way. Here, we have seen that
the ABE-based approach is feasible to send a large amount of data efficiently. Our second comparison
evaluated the detectability of non-permitted datasets. The ABE-based approach here offers the a-priori
advantage that a customer can check if he is able to fulfill a policy before he tries to decrypt
the appropriate ciphertext. Our evaluation confirmed this assumption. Lastly, we found out that the ABE-
based approach can compensate its initialization after a few access processes. In the corresponding
test, the initialization and multiple access processes were executed. We saw that the ABE-based
approach outperformed the PDS-based approach after approximately 250 access processes. That is,
according to Section 2.3, not even enough for one customer to get one representative result.

Through the gained knowledge, our thesis goals have been fully achieved. We have not only shown
that a Data Exchange Platform can be implemented with full respect to privacy aspects and without a
managing third party that can monitor the entire data flow. We have even implemented two
completely different approaches. Both offer much better privacy aspects than the status quo approach
and even a very good performance. Especially the ABE-based approach fulfilled our requirements in a
great measure. The concept of combining access control with cryptographical functioning pays off very
well for our use case of a Data Exchange Platform.

However, our results still leave us with some open questions. We have seen that the policy length has
a higher impact on the performance than the policy height, but according to the current state of
science, there is no optimizer that can improve policies in this respect. Due to the \(\text{coNP}\) hardness of
the optimization problem, we even have to assume that a perfect optimizer for this purpose will never
exist. However, strategies for simplification are still desirable. We also discussed possibilities to make
the PDS-based approach more feasible but found out that a generally better performance is not
possible without neglecting more privacy aspects.

In summary, we must say that both approaches are important privacy enhancement tools. The
previous examples on the PDS-based approach show, that it offers the advantage of a good overview
of datasets and determined access rights. However, the ABE based approach was more suitable for our
use case of a Data Exchange Platform which required a fast and automated system. Through its
approach of treating datasets as individually and encapsulated from each other, it fulfills important
functional requirements and also delivers a strong performance. Relating to our use case, the ABE-
based approach allows a strong implementation of a Data Exchange Platform in which vendors and
customers can act with full consideration of the LINDDUN privacy threats and general security threats
so that they can be sure that their data is adequately protected at all times.
10 Publication Bibliography


Deng, Mina; Wuyts, Kim; Scandariato, Riccardo; Preneel, Bart; Joosen, Wouter (2011): A privacy threat analysis framework: supporting the elicitation and fulfilment of privacy requirements. In Requirements Engineering, pp. 3–32.


Goyal, Vipul; Pandey, Omkant; Sahai, Amit; Waters, Brent (2006): Attribute-based encryption for fine-grained access control of encrypted data. In CCS ’06 Proceedings of the 13th ACM conference on Computer and communications security, pp. 89–98.


Grossklags, Jens; Acquisti, Alessandro (2007): When 25 Cents is too much: An Experiment on Willingness-To-Sell and Willingness-To-Protect Personal Information. In WEIS.


Mayer-Schönberger, Viktor; Cukier, Kenneth (2013): Big data: A revolution that will transform how we live, work, and think. In Redline Wirtschaft.


Appendix A

The previous scheme, including a security proof, was provided by Bethencourt et al. [Bethencourt et al. 2007, S. 326–327]. Before we proceed, we want to show, why two malicious parties that join for a collusion attack cannot abuse the scheme to join their keys $SK$ and $\bar{SK}$ to decrypt $CT$ with $\text{Decrypt}(CT, SK) = \text{Decrypt}(CT, \bar{SK}) = \bot$. We assume, that there is a non-leaf node $x'$ and its successors $C_{x'}$ with $\text{DecryptNode}(CT, SK, x') = \text{DecryptNode}(CT, \bar{SK}, x') = \bot$, but $\text{DecryptNode}(CT, SK', x') \neq \bot$ where $SK'$ is the output of $\text{KeyGen}(MK, S \cup \bar{S})$. To simplify our sketch, we assume, that $S_x$ lacks exactly one attribute to decrypt $CT$, that is the leaf $z'$.

**Assumption:** There is no function $\text{DecryptNode}(CT, SK, \bar{SK}, x) \neq \bot$, based on the CP-ABE scheme which makes a collusion attack possible.

**Proof:** We consider, what a try of the two parties, to use the union of their keys as input for $\text{DecryptNode}$ results in:

$$\prod_{z \in C_{x'}} F_{\Delta z^i(0)}^{S_{x'}^i}, \text{where } i = \text{index}(z), S_{x'}^i = \{\text{index}(z): z \in S_x \cup \bar{S}_x\}$$

$$= \prod_{z \in C_{x'} \cap S_x} (e(g, g)^{r \cdot q_p(\text{index}(z))})^{\Delta z^i(0)} \cdot (e(g, g)^{r \cdot q_p(\text{index}(z))})^{\Delta z^i(0)}$$

$$= \left( e(g, g)^{\Delta z^i(0)(r \cdot q_{\text{parent}(z')}(\text{index}(z_1))) + \ldots + r \cdot q_{\text{parent}(z')(\text{index}(z_{|C_{x'}|-1}) + \cdot r \cdot q_{\text{parent}(z')(\text{index}(z_{|C_{x'}|}))}. \right)$$

The problem now is, that we cannot apply the polynomial interpolation that leads us to a uniform exponent, since one of the nodes is blinded by a random parameter $\tilde{r}$, while all the others are blinded by $r$. Without a uniform exponent, however, we cannot identify, which part of the exponent belongs to $s$ (remark: $rs = r \cdot q_x(0)$ was the result in the exponent at the correct encryption). Without that strict division to $r$ and $s$, however, we cannot unblur (divide) the factor $r$ in $D = g^{(\alpha + r')/B}$ so that we are unable decrypt the ciphertext.
Appendix B

For dynamic testing purposes, we have implemented a parser which can be used to set up and simulate a variable Data Exchange Platform. The parser is implemented and can be executed from the ParserTest.java class. We present the commands with which the parser can be used and the parameters. The parameter \(-h\) shows all needed parameters for each command.

createCustomer: Creates an arbitrary number of customers and assigns IDs in ascending order, starting at 1.
\(-n:\) Number of created customers

listCustomer: Lists the IDs of all currently existing customers

createVendor: Creates an arbitrary number of vendors and assigns IDs in ascending order, starting at 1.
\(-n:\) Number of created vendors

listVendor: Lists the IDs of all currently existing vendors

registerDataset: Registers a data domain (e.g. favorite movie) and assigns an ID in ascending order, starting at 1.
\(-d:\) Description of the generated data domain

listDataset: Lists the IDs of all currently existing data domains and their descriptions

createDataPDS: Creates a dataset in a vendor’s Personal Data Store
\(-v:\) ID of the vendor who generates the dataset
\(-d:\) ID of the corresponding data domain
\(-l:\) List of all customers who are allowed to access the generated dataset
\(-e:\) Information / Entry that is stored in the dataset

createInterfacePDS: Creates an Interface in a vendor’s Personal Data Store for a customer
\(-v:\) ID of the vendor who created the interface
\(-c:\) ID of the customer who receives the access rights
\(-l:\) List of the datasets that are approved for the customer

getDataPDS: Requests a dataset in a vendor’s Personal Data Store from a customer’s point of view
\(-v:\) ID of the vendor who owns the Personal Data Store
\(-c:\) ID of the customer who requests the dataset
\(-d:\) Data domain of the requested dataset

createAttributeABE: Creates an attribute as needed for the ABE-based approach
\(-t:\) Title of the attribute
\(-d:\) Meaning/Description of the attribute

getAttributeID: Lists the IDs of all currently existing attributes

provideAttributeABE: Assigns an attribute to a customer in ascending order, starting at 1.
\(-c:\) ID of the receiving customer
\(-a:\) ID of the attribute
createDataABE: Creates a dataset in a centralized ABE-based approach
   -v: ID of the creating data vendor
   -d: ID of the data domain
   -e: Information / Entry that is stored in the dataset
   -p: The policy under which the dataset is protected

getDataABE: Requests a dataset from a customer’s point of view
   -c: ID of the requesting customer
   -v: ID of the vendor who created the dataset
   -d: ID of the requested data domain

end: Ends the Parser