# Data Protection in the Cloud

Jan de Muijnck-Hughes BSc (Hons)

March, 2011

| | |
|---|---|
| s0819824 | **Student No** |
| 640 | **Thesis No** |
| prof. Bart Jacobs and dr. Erik Poll | **Supervisor(s)** |

# Data Protection in the Cloud

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

specialising in

COMPUTER SECURITY RESEARCH

by
Jan de Muijnck-Hughes BSc (Hons)
j.demuijnck-hughes@student.ru.nl

Digital Security Group,
Institute for Computing and Information Sciences,
Faculty of Science,
Radboud University Nijmegen,
Nijmegen, Gelderland,
The Netherlands
http://www.ru.nl/ds

## Abstract

Cloud Computing sees a technical and cultural shift of computing service provision from being provided locally to being provided remotely, and en masse, by third-party service providers. Data that was once housed under the security domain of the service user has now been placed under the protection of the service provider. Users have lost control over the protection of their data: *No longer is our data kept under our own watchful eyes.*

This thesis investigates how Predicate Based Encryption (PBE) could be leveraged within the Cloud to protect data. PBE is a novel family of asymmetric encryption schemes in which decryption of a cipher-text is dependent upon a set of attributes satisfying a certain predicate, allowing for selective fine-grained access control to be specified over cipher-texts.

It is argued that obfuscation of one's data is not enough when seeking to protect data. The control of how one's data is used and the trust afforded to service providers is equally as important. To this end, three archetypal scenarios are described that illustrate ways in which service users could specify precisely with whom they wish to share their data, for what purpose, and for how long. Furthermore, two additional scenarios are presented that would allow a service provider to facilitate keyword search over encrypted data using expressive queries supporting conjunction and disjunction of terms.

# Declaration

I declare that the material submitted for assessment is my own work, except where credit is explicitly given to others by citation or acknowledgement.

In submitting this project to the *Radboud Universiteit Nijmegen*, I give permission for it to be made available for use in accordance with the regulations of the *Radboud Universiteit Nijmegen*. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied to any bona fide library or research worker, and to be made available on the World Wide Web.

Jan de Muijnck-Hughes BSc (Hons)
March, 2011

To-morrow, and to-morrow, and to-morrow. [...] It is a tale, told by an idiot, full of sound and fury, signifying nothing.

Macbeth (Act V, Sc. 5)

# Contents

# Mathematical Notation

| Symbol | Term | Description |
|---|---|---|
| $\mathcal{A}$ | Access Structure | A data structure that describes sets of authorised items. |
| $\{\vert M\vert\}_K$ | Asymmetric Cryptography | Represents asymmetric cryptographic operations. |
| $\mathcal{S}$ | Attribute Set | Represents a set of attributes from $\mathcal{U}$ |
| $\mathcal{U}$ | Attribute Universe | Represents the universe of all attributes used. |
| $CT$ | Cipher-text | Information that has been encrypted into a ‘meaningless’ form. |
| $\mathsf{Dec}(Entity)$ | Decryption Key | Decryption (private) key of an Entity |
| $\mathsf{Enc}(Entity)$ | Encryption Key | Encryption (public) key of an Entity |
| $\mathcal{H}$ | Hash | Well defined procedure to convert arbitrary data into a fixed-size bit string. |
| $ID_{entity}$ | Identity | Represents the identity of an entity. |
| $\mathsf{MPK}$ | Master Public Key | Public parameters of a crypto-system. |
| $\mathsf{MSK}$ | Master Secret Key | Private global parameters of a crypto-system. |
| $M$ | Message | Unencrypted information that is readable to all. |
| $(\mathcal{M}, \rho)$ | Monotone Span Program | A data structure used to describe any access structure. |
| $A \rightarrow B : M$ | Send Message | Send a message $m$ from Entity $A$ to Entity $B$. |
| $\{M\}_K$ | Symmetric Cryptography | Represents symmetric cryptographic operations. |

Chapter 1

# Introduction

Perfer et obdura; dolor hic tibi proderit olim.

Ovid

## 1.1 Motivation: Alice and her Suitors

The use of encryption schemes is often described through an analogy depicting the transmission on a plain-text message $M$ from one entity, *Alice* to another entity, *Bob*. Here Alice wishes to ensure that only Bob will be able to read $M$. This analogy has persisted due to its ability to describe a prevalent communication style, that of *unicast* communication. However, this simple analogy does not necessarily represent the entirety of communication styles that are actively used, it does not take into account *multicast* communication: *What if Alice's wish were to send her message not to Bob but to Bobs plural?*

Traditional symmetric and asymmetric encryption schemes can be leveraged to provide Alice with a secure means through which she can send her message. However, with symmetric schemes each recipient will be in a position to decrypt *all* cipher-texts that have been encrypted with the same key: *Access is too coarse-grained.* With asymmetric schemes the encrypting entity needs to explicitly state for whom decryption is permissible: *Access is too fine-grained.* To reference the different styles of communication, symmetric schemes represent *broadcast* communication and asymmetric schemes *unicast* communication. A *multicast* encryption scheme is required that allows for a more expressive fine-grained means through which Alice can specify access over her data.

A promising solution to 'multicast' encryption is that of Predicate Based Encryption (PBE). PBE is a novel family of asymmetric encryption schemes in which decryption of a cipher-text is dependent upon a set of attributes satisfying a predicate. A generalisation of both Attribute Based Encryption (ABE) [SW05] and Identity Based Encryption (IBE) [Sha85], PBE allows for selective fine-grained access control to be specified over encrypted data. Generally speaking, attributes are used to construct users' decryption keys and to encrypt plain-text messages. Decryption occurs when a match occurs between the attributes held by the entity (in their decryption key) and the attributes used to construct a cipher-text. This matching occurs through the use of predicates, that describe: a) the required attributes needed to decrypt; and b) the relationship between the attributes.

## 1.2 Cloud Computing

*Cloud Computing* is the name given to the recent trend in computing service provision. This trend has seen the technological and cultural shift of computing service provision from being provided *locally* to being provided *remotely* and *en masse*, by third-party service providers [Hay08]. Functionality such as storage, processing and other functionality is now offered on demand, *as a service* and both freely and at cost [AF+09]. Data, that was once housed under a consumers own administrative and security domain, has now been extracted and placed under the domain of the Cloud Service Provider (CSP) [Pea09]. The consumer has effectively lost control over how their data is being stored, shared and used, and also over the security used to protect their data. Moreover, it can be the case that a surreptitious employee of the service provider will have access to your data for legitimate purposes but will abuse this power for their own means [Won10]. Users are no longer in full control over the security of their data and the protection offered by the service provider is not absolute. There is a need for users to have more control over the protection of their data within the cloud: *Users need to become empowered.*

## 1.3 Research Project and Objectives

Given service users' need to regain control over their data together with PBE's ability to offer selective fine-grained access control over encrypted data. This thesis seeks to investigate:

How Predicate Based Encryption schemes can be leveraged within the Cloud to protect data.

The investigation was divided broadly into three stages.

- *Data Security and the Cloud.* The initial stage sought to provide a clear definition for Cloud Computing and the security issues therein, looking to identify precisely where and when threats can occur to data and how these threats ought to be mitigated.

- *Predicate Based Encryption.* The next stage focused solely upon PBE schemes discussing how they work and what they allow for. This provided a foundation upon which their deployment as part of a crypto-system could be explored and to define the types of problem that PBE schemes can be used to solve.

- *Leveraging PBE.* The final stage of the investigation built upon, and combined the results, of the previous stages. Here the investigation looked to determine the problems that PBE schemes can be used to solve within the Cloud, and the quality of solution provided.

## 1.4 Research Outcomes

From the investigation, it was determined that PBE can be used to protect data within the cloud. The main results for each stage of the investigation are outlined below.

- *Data Security and the Cloud.* From the initial stage two threat-models were produced: one user, and the other CSP orientated. These models described the threats upon data in terms of the data lifecycle. Furthermore, it was determined that a privacy model centred around Kafka's *The Trial* together with the idea that the CSP provider could be trusted facilitates a better understanding of the problems present within the Cloud and how such problems can be solved.

- *Predicate Based Encryption.* Characteristics that can be used to categorise PBE schemes were identified. Of which predicate placement had the greatest affect upon the access control afforded by the scheme. A generic model for deploying PBE schemes as part of a crypto-system was developed. From this model three modes of operation that characterises the deployment of a PBE scheme were identified.

- *Leveraging Predicate Based Encryption.* Three scenarios are described that illustrate ways in which service users could specify precisely with whom they wish to share their data, for what purpose, and for how long. Furthermore, two additional scenarios are presented that would allow a service provider to facilitate keyword search over encrypted data using expressive queries supporting conjunction and disjunction of terms.

## 1.5 Organisation

**Part I: Data Security and the Cloud** Within the first part of the thesis a definition towards Cloud Computing is given (Chapter 2) together with an overview of the technical and legal security issues found therein—Chapter 3. Chapter 4 introduces two threat models that establish what threats can occur to data in the Cloud and where. Several security requirements that govern data in the Cloud are discussed in Chapter 5. Finally this part concludes with Chapter 6 that provides a discussion over the trappings of what makes a good solution.

**Part II: Predicate Based Encryption** The second part of this thesis introduces PBE, the construction of PBE schemes, and PBE schemes use as part of a cryptographic system. As with any modern encryption scheme the mathematics involved can be complex and intimidating for those not already versed in the field. Chapter 7 provides a high-level overview towards PBE schemes and their operation. For those more versed in mathematics, Chapter 8 looks at the underlying mathematics surrounding the construction of PBE schemes that use general predicates. Chapter 9 discusses the use of PBE as a part of a cryptographic system. Finally, Chapter 10 provides an evaluation over PBE.

**Part III: Leveraging PBE in the Cloud** The third part of this thesis discusses how PBE could be leveraged within the Cloud. Chapter 11 introduces five scenarios and Chapter 12 evaluates them.

**Part IV: Conclusion, Reflection, and Further Work** Chapter 13, provides a summary over the research findings and Chapter 14 provides information pertaining to future directions that this topic could be taken in and over related areas of interest.

# Part I

# Data Security within the Cloud

Chapter 2

# Defining Cloud Computing

*A definition for Cloud Computing is given together with an overview concerning computing as a service and the benefits from using the Cloud.*

## 2.1 Overview

*Cloud Computing* is the name given to a recent trend in computing service provision. This trend has seen the technological and cultural shift of computing service provision from being provided *locally* to being provided *remotely* and *en masse*, by third-party service providers [Hay08]. These third-parties offer consumers an affordable and flexible computing service that consumers would otherwise not have been accessible, let alone afford [MKL09, Chapter 1]. This new means of service provision has evolved from and is the culmination of research stemming from (among others) distributed and networked systems, utility computing, the web and software services research [Vou08; AF+09]. This paradigm shift has led to computing being seen as another *household utility*, aka "fifth utility", and has prompted many a business and individual to migrate parts of their IT infrastructure to the cloud and for this data to become managed and hosted by Cloud Service Providers (CSPs). However, Cloud Computing is the *cause celébré* among tech pundits and has led to the term 'Cloud Computing' as an umbrella term being applied to differing situations and their solutions. As such a broad range of definitions for Cloud Computing exists, each of which differ depending on the originating authors' leaning. This chapter seeks to provide a coherent and general introduction to Cloud Computing.

## 2.2 Computing as a Service

One of the main tenets of Cloud Computing is the 'as-a-Service' paradigm in which 'some' service is offered by a *Service Provider* (also known as a Cloud Service Provider) to a *User* (consumer) for use. This service can also be categorised according to the application domain of its deployment. Examples of application domains that offer services are: Financial e.g. Mint.com, Managerial e.g. EverNote and Analytical e.g. Google Analytics. The agreed terms of use, indicating the actions that must be taken by both the provider and consumer, are described in a contract that is agreed upon before service provision. Failure to honour this agreement can lead to denial of service for the consumer or legal liability for the service provider. This contract is often described as a *Terms of Service* or *Service Level Agreement*. Moreover, as part of this agreement the service provider will provide a *Privacy Policy* which outlines how the users data will be stored, managed, used and protected.

### 2.2.1 Service Levels

The services offered are often categorised using the *SPI Service Model*. This model represents the different layers/levels of service that can be offered to users by service providers over the different application domains and types of cloud available [MKL09, Chapter 2]. Clouds can be used to provided as-a-Service: software to use, a platform to develop on, or an infrastructure to utilise. Figure 2.1 summarises the *SPI Service Model*.

**Software as a Service** The first and highest layer is known as: Software as a Service (SaaS). It represents the applications that are deployed/enabled over a cloud by CSPs. These are mature

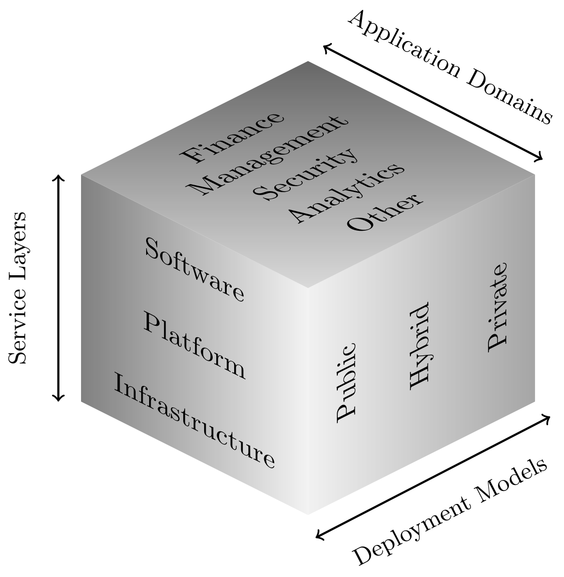


Figure 2.1: Summary of the *SPI* Service Model

applications that often offer an API to allow for greater application extensibility. For instance, Google Docs can be seen as the archetypal SaaS application, it has been deployed solely within the Cloud and offers several APIs to promote use of the application.

**Platform as a Service** The next layer is known as: Platform as a Service (PaaS). This represents a development platform that developers can utilise to write, deploy and manage applications that run on the cloud. This can include aspects such as development, administration and management tools, run-time and data management engines, and security and user management services. For instance, *Force.com* and *Amazon Web Services* [AWS] offers a suite of services that allows developers to construct an application that is deployed using web-based tooling.

**Infrastructure as a Service** The final and lowest layer is known as: Infrastructure as a Service (IaaS). CSP offer developers, a highly scaled and elastic computing infrastructure that are used to run applications. This infrastructure can be comprised of virtualised servers, storage, databases and other items. Two well known examples are the *Amazon Elastic Compute Cloud*, a commercial platform offered as part of *Amazon.com*'s Web Service platform and *Eucalyptus*, an open source platform that offers the same functionality [AWS; NW+09].

### 2.2.2 Entities Involved

Cloud actors/entities can be divided into two main categories: A) CSP or Service Provider those who provide a service; and B) *Cloud Service User* (Users) those who use a service. Within Cloud Computing the differences between the role played by a service provider and a user can be blurred. The service provider could also be the user of another service e.g. when infrastructure is the service. The exact definition whether an entity is a provider or user is dependent on the context of the interaction and the service being offered. Some service providers will offer services at all three service levels, or offer just one particular level of service and have their own internal IaaS infrastructure.

A possible refinement could be that CSP providers are either: a) *Infrastructure Service Providers*—those that offer IaaS and own and run the data centers that physically house the servers and software; or b) *Service Providers*—those that offer PaaS or SaaS services. And that*Cloud Service Users* are either: A) *Platform Users*— are users who buy into a service providers platform e.g. Facebook; and B) *Consumers*—are service users who use either SaaS or IaaS services.

## 2.3 Defining the Cloud

The term ‘cloud’ has been used traditionally as a metaphor for networks and helps abstract over their inherent complexity. This term, however, has evolved to encompass the transparency between the technological infrastructure of the CSP and the consumers point of view. A cloud can be one of the following types:

**Public** Constituting publicly accessible services that are accessed over the Internet and are often described using the term *“The Cloud”*.

**Private** These are private services deployed on private networks. Such clouds may also be managed by third parties.

**Hybrid** A combination of services offered both privately and publicly. For example core-services may be offered on a private cloud; other services originate from public clouds.

Vaquero, Rodero-Merino et al. provides a definition for Clouds based upon the commonalities found among existing definitions [VRM+09]. They define Clouds to be:

> *“. . . a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs.”*

The provision of virtualised resources, as a service, allows for an elastic and on demand computing environment that can be expanded and shrunk as needed, depending on the needs of the consumer. Mather, Kumaraswamy and Latif [MKL09, Chapter 2] provides a slightly alternate definition. This definition is based on five attributes that can be used to describe a cloud-based system. They are:

**Multi-tenancy** The sharing of resources being offered by the service providers to the consumers at the network, host and application level.

**Massive Scalability** The ability to scale the storage, bandwidth and systems available to proportions unattainable if performed by the organisation itself.

**Elasticity** Consumers can rapidly and dynamically increase and decrease the resources required as needed.

**Pay-as-you-go** Consumers only pay for the resources they consume such as processing cycles and disk space.

**Self-Provisioning of Resources** Consumers have the ability for the self-provisioning of resources.

This is a better definition, it does not pertain to a particular technology. However, one attribute **Pay-as-you-go** is in itself too constrictive as it pertains to a particular style of payment and prohibits subscription based models. A better attribute should be:

**Flexible Payment** Payment for resource usage is flexible and consumers can be offered different payment models dependent on their intended use of the resources.

For instance in Amazon EC2 pricing for image use is dependent on three factors: 1. *Image Type*—On Demand, Reserved, Spot 2. *OS Used*—UNIX/LINUX or Windows; and 3. *Data Center Location*—West Coast America, East Coast America or Ireland. Consumers are billed differently per hour based upon these factors and other services used [AWS-Pricing]. On the other-hand Google Apps Premium Edition will cost companies $50 per user per year for their IT infrastructure solution [GooApps] and users will have a fixed set of resources. The use of such models often ensures for a lower operational cost than when compared with an in-house solution.

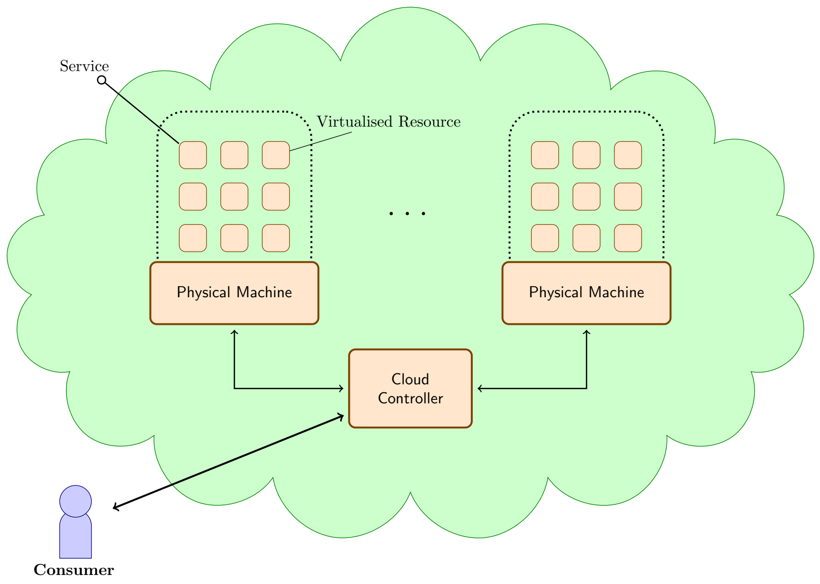


Figure 2.2: The Data Center Model for Cloud Computing.

When discussing the physical constructions of Clouds the *Data Center Model* is a popular choice. This model stipulates clusters of machines running specialist, dedicated hardware and software to provide the infrastructure for the large scale provision of *virtualised* resources [AWS; NW+09]. Though a consumer can access the virtualised resources directly, when managing their resources, consumers interact with a *Cloud Controller* that mediates the interaction between the consumer, and: a) the physical machines; and b) the virtualised resources owned by the consumer. Figure 2.2 illustrates a rather simplified view of this data center model. A more interesting realisation of clouds is the *Adhoc* model in which the existing (idle) computing power within an enterprise is tapped to provide the required infrastructure. The more interested reader is asked to read Kirby, Dearle et al. [KD+09] for more information concerning the adhoc cloud model.

## 2.4 Benefits of Cloud Computing

Many of the benefits to be had when using Cloud Computing are the lower costs associated. At the infrastructure level, virtual images can be scaled and contracted with complete disregard for any associated hardware costs such as equipment procurement, storage, maintenance and use. This is all taken care of by the service provider and will be factored into the payment for the service: capital expenditure has been converted into operational expenditure. Resources within the cloud can be treated as a commodity, an 'unlimited' medium. At both the platform and software level similar benefits are seen. Aspects such as software installation, deployment and maintenance is virtually non-existent. This is taken care of by the provider within their own infrastructure. The service user only pays technical support.

Service providers at the SaaS level, often *tout* features that allow users to collaborate and interact with each other, in real-time, within the scope of the service being offered. For example, Google Docs allows users to edit documents simultaneously and for users to see each others edits in real time. Moreover, the provision of platform and software 'as a service' allows cloud service users the ability to aggregate services together either for their own use or to promote as another service i.e. *Mashups*. The aggregation could imply the combination of functionality from several services, or the change/combination of output from the services involved.

**Remark.** *Service aggregation is a good example outlining how a service user can become a service provider.*

Chapter 3

# Cloud Security Issues

*Security issues to be found within the cloud from both a technical and socio-technical perspective are discussed.*

## 3.1 Overview

Security issues come under many guises both technical and socio-technical in origin. To cover all the security issues possible within the cloud, and in-depth, would be herculean—a task not suited even for Heracles himself. Existing efforts look to provide a taxonomy over the issues seen. The *Cloud Security Alliance*[1] is a non-profit organisation that seeks to promote the best practises for providing security assurance within the cloud computing landscape. In Hubbard, Sutton et al. [HS+10] the Cloud Security Alliance identify seven threats to cloud computing that can be interpreted as a classification of security issues found within the cloud. They are:

1. Abuse and Nefarious Use of Cloud Computing
2. Insecure Application Programming Interfaces
3. Malicious Insiders
4. Shared Technology Vulnerabilities
5. Data Loss/Leakage
6. Account, Service and Traffic Hijacking
7. Unknown Risk Profile

For an alternate classification of threats to Cloud Computing, one can consult [ENISA-CC-RA09]. This chapter provides a general view of the security issues within the scope of the threats presented by the Cloud Security Alliance. This chapter concludes with an overview of the legislative aspects governing security within the cloud.

## 3.2 Abuse and Nefarious Use of Cloud Computing

Legitimate CSPs can be abused for nefarious purposes, supporting criminal or other untoward activities towards consumers. For instance, services can be used to host malicious code or used to facilitate communication between remote entities i.e. botnets. The emphasis is that legitimate services are used with malicious intent in mind. Other issues seen include the provision of purposefully insecure services used for data capture.

Service providers may entice potential users with offers too good to be true. For example the promise of unlimited resources or a '30-day Free Trial'. During the registration process the consumer will be asked to provide more information than what would normally be required under the pretence of providing service personalisation e.g. location or age based advertisement. Commonly asked information includes the consumers name, email/postal address, D.O.B or even credit card details. Users are essentially being goaded to part with more information than required as a prerequisite for service use. Malicious entities can then use this information

[1] `http://www.cloudsecurityalliance.org/`

for nefarious purposes, most notably identity theft. Even if the entities are not malicious the disclosure of information to the CSP can also be said to be an abuse of service by the CSP themselves. CSPs may collect this information, or any other information provided at later stages, and market this information to third parties for data mining purposes. This is akin to store cards such as *Tesco Clubcard* in which the perceived benefits i.e. monetary savings, come at the cost of handing over personal information such as shopping habits.

Another security issue found is the provision of legitimate maliciously-oriented services. This service provision. also known as *(In)Security-as-a-Service*, offers users the same security guarantees as existing services yet their use is malicious. For example `http://www.wpacracker.com/` is a cloud-based cracking service that can be used to 'check' the security of WPA-PSK protected wireless networks. By using a cloud based service it is claimed that a process that would take around five plus days now takes on average twenty minutes.

## 3.3 Insecure Interfaces and Application Programming Interfaces

Data placed in the Cloud will be accessed through Application Programming Interfaces (APIs) and other interfaces. Malfunctions and errors in the interface software, and also the software used to run the Cloud, can lead to the unwanted exposure of users data and impugn upon the data's integrity. For example a (fixed) flaw in Apache, a popular HTTP server, allowed an attacker to gain complete control over the web server [Ho10]. Data exposure can also occur when a software malfunction affects the access policies governing users data. This has been seen in several Cloud based services in which a software malfunction resulted in which a users privacy settings were overwritten and the user data exposed to non-authorised entities [For10; Vas09]. Threats can also exist as a result of poorly designed or implemented security measures. If these measures can be bypassed, or are non-existent, the software can be easily abused by malicious entities. Regardless of the threat origin, APIs and other interfaces need to be made secure against accidental and malicious attempts to circumvent the APIs and their security measures.

## 3.4 Malicious Insiders

Although a CSP can be seen as being honest their employees may not be. A malicious insider is an employee of the CSP who abuses their position for information gain or for other nefarious purposes e.g. disgruntled employee. Regardless, of the employees motivation the worrying aspect is that a surreptitious employee will have access to consumers data for *legitimate purposes* but will abuse this power for their own means [Won10].

Another more subtle form of the malicious insider problem is through PaaS based services. If the service provider offers a platform that allows developers the ability to interact with users data i.e. Facebook Applications, users may unknowingly allow these developers access to all their data. Use of this platform may be unchecked. For example, it is well known on the Facebook Platform that once a user adds an application the application will have the ability to access all the users information, if allowed to do so, regardless of the applications function. With the popularity of these applications it raises the question of: *How safe are Cloud based applications*? Even if the application developers are not malicious this does not mean that the application cannot be hacked [Tho09; ACLU-fb-app; Per09].

## 3.5 Shared Technology Issues

A more interesting form of confidentiality issue relates to the construction of a cloud and the services themselves.

### 3.5.1 Virtualisation Issues

The underlying virtualisation architecture allows IaaS service providers the ability to host several machine images on a single server. Ristenpart, Tromer et al. [RT+09] discuss practical attacks on such services, concentrating on Amazon EC2. First, the authors showed that they could map the internal structure of the cloud, allowing them to determine if two virtual machines were co-resident with each other i.e. were running on the same physical machine. Secondly, they demonstrated that they were able to, purposefully, add a virtual machine to the cloud so that it was co-resident with another machine. Finally, the authors were able to show that once a machine was co-resident, they would be able to launch several attacks that would allow them to learn information regarding CPU cache use, network traffic rates and keystroke timings.

### 3.5.2 Service Aggregation

Aggregated services offer services based upon the functionality offered by existing services. Often aggregated services offer the combined functionality of existing services allowing for rapid service construction. However, service aggregation presents consumers with several interesting problems [Pea09]. Data is now being shared across multiple service providers whose privacy policies will also subject to change. Under whose privacy policy is the data governed by, how to combine the two policies? Furthermore, service aggregation can occur in an ad-hoc and rapid manner implying that less stringent controls could have been applied to the protection of data, increasing the likelihood of a problem.

## 3.6 Data Loss or Leakage

Although insecure APIs can lead to data loss or the unwanted exposure of information, consumers can also loose their information through other means.

### 3.6.1 Availability Issues

Availability issues are when users data is made inaccessible to the consumer. The data has been made unavailable. Such a lack of availability can be a result of access privilege revocation, data deletion or restricting physical access to the data itself. Availability issues can be attributed to an attacker using flooding based attacks [JGL08]. For example, Denial of Services attacks, attempt to flood the service with requests in an attempt to overwhelm the service and cease all of the services intended operations.

A monetary effect can also be seen with availability issues. Monetary issues affect not only the consumer but also the CSP. Recall from Section 2.3, that one of the benefits of Cloud Computing is that of *Flexible Payment*, consumers can be charged on a pay-as-you-go tariff. Flooding attacks will have an effect upon resource utilisation and will result in increases to power consumption, network usage and hardware maintenance. Ultimately this will also increase the amount of money the consumer will be charged for resource usage. Moreover, these monetary increases will also increase the operational expenditure of the service provider [JS09].

Fault tolerance protocols are used to combat the issue of node failure within the Cloud [Cri91]. Fault tolerance protocols replicate data across machines, and data centers, ensuring that if part of the cloud does fail a version of the data will still be available to the user. Part of the cloud will be redundant though for good measure. Data replication also requires that the data state of the data be synchronised across multiple nodes. However, poorly designed protocols can also lead to availability issues. Birman, Chockler and Renesse [BCR09] discuss that synchronisation protocols with tightly coupled nodes have more adverse affects such as higher network usage and deadlock when compared to loosely coupled asynchronous nodes. Furthermore, the existence of multiple copies of data can also introduce confidentiality problems. The increased number of data instances also increases the likelihood that an attacker will be able to access the data.

### 3.6.2 Data Leakage

Another form of data leakage stems from the disclosure of information that, though hidden, is deduced from freely available information. For example: say that Bob is a member of a rather masculine society i.e. rugby club. Say that Bob is also homosexual and is actively involved with an LGBT society. Bob may wish to keep his sexuality a secret from his friends in the rugby club due to fears of being emasculated. In this situation Bob will wish to keep his on-line activities involving these two facets of his private life separate. However, this does not mean that *never the twain shall never meet.* In Chew, Balfanz and Laurie [CBL08] the authors discuss ways in which the privacy of social networking sites can be undermined. The authors describe how such personal information e.g. Bob's sexuality, can be disclosed from *unwelcome linkage* and also through *social graph merger*.

When users interact with a service they can leave a public trail, be it from status/update messages or through new postings. Unwelcome linkage occurs when new information is discerned about an individual through analysis of the individuals public trail i.e. links. This unwelcome linkage could be accidental or the result of the individual not covering their tracks. That is Bob may keep a private, anonymous blog detailing his frustration with being homosexual in the rugby club and use a photo from an online photo account that is linked to his real identity. This is *accidental linkage*. Similarly, a friend of Bob's from the LGBT society could place a link to Bob's blog within their own blog and refer to Bob's real identity. This is defined as *trackback linkage* from Chew, Balfanz and Laurie [CBL08].

Social graph merging is similar to unwelcome linkage however the links formed occur through the aggregation of social graphs. A social graph is a graph describing a person's social information such as friends, groups and interests.

Through combination of a persons social graphs from separate social networking sites, or the social graphs of people from the same social network site, new information can be deduced. Continuing with the example of Bob used earlier, it is feasible that through the analysis of Bob's social graph, and the social graphs of his friends, Bob's sexual orientation can be deduced. This attack to determine the sexual orientation of a person and the ability to 'out' them was shown to be feasible in Jernigan and Mistree [JM09]. The authors determined that the more homosexual friends that an individual has the higher the probability that the person was also homosexual. Furthermore, both Narayanan and Shmatikov [NS09] and Wondracek, Holz et al. [WH+10] demonstrated practical attacks that analyse a persons social graph and allow the persons identity to be revealed i.e. de-anonymised.

## 3.7 Account or Service Hijacking

When communicating with the CSP malicious entities may seek to affect the integrity and authenticity of the user's communication with the CSP and *vice versa*. There are several ways in which the integrity and authenticity of a users session can be impugned [JS+09, Section 3.1–2]. The rise of 'Web 2.0', has seen the web browser becoming increasingly used as a means to access remote services. Browser-based interfaces and authentication are used by consumers to establish a session with their service provider. A malicious entity can attempt to capture or hijack this session or steal the users credentials to access or influence the users data, from within the browser. Most browsers operate on a *Same Origin Policy* where client scripts are allowed to access resources if they share the same origin. However, attacks such as *Cross Site Scripting* [XSS02] and *Cross Site Request Forgery* [XSRF10], as well as DNS Poisoning [Lem08] can be used to undermine this security feature. Other issues found include the manipulation of the data being sent within the sessions. For example the *XML Signature Element Wrapping* attack [MA05] targets protocols that make use of WS-Security. The attacker will capture an XML-Signed SOAP message and modify it such that the original body is placed within the SOAP header and in its place is the malicious payload. When checked by the recipient the original signature will still hold.

The effects of breaking session integrity are two-fold, for one the attacker will be able to steal the identity of their victim, and secondly impugn the reputation of the victim through falsified data. Such *man-in-the-middle* attacks will have lasting repercussions such as violation of the services terms of use, or criminal. For instance, the hacking of many a celebrities twitter account [Arr09].

**Remark.** *The man-in-the-middle attack is also an affront to the confidentiality of data if the attacker is able to read the data as well as modify it.*

## 3.8 Unknown Risk Profile

Risk Management is a business process that users can use to identify and mitigate threats. It allows users to determine their current stance towards the security of their data. Auditing information such as software version, code updates, current security practises, intrusion attempts *et cetera*, are used as a basis for determining this stance. CSPs may not be so forthcoming with this information. Consumers, when adopting a service, must also accept the *Terms and Conditions* (including privacy policy) of the service, together with any *Service Level Agreements* made. Consumers and providers need to comply with existing laws and regulations. However the degree to which service providers adhere to current security practises and legislation, or implement them may not be clear. This leaves the consumers with an unknown risk profile. Users are unable to determine the risk to their data as they do not have sufficient information to do so.

## 3.9 Jurisprudence Oriented Issues

Jaeger, Lin and Grimes [JLG08] discusses the widening gap between technology and local, national and international legislation. Technological innovation occurs at a much more rapid pace than the pace at which legislation can change. This rapid pace has left a cloud of ambiguity concerning how users' data can be treated legally within the cloud. Given the nature for 'ever-changing' service level agreements, the terms and conditions that one originally agreed to may not be the current versions. Furthermore, the laws and regulations themselves may not be suitable, are open for interpretation and also bounded by jurisdiction. These issues are discussed below.

### 3.9.1 Service Level Agreements

Users implicitly trust the service provider not to violate terms and conditions, and be in a position to securely store their information. The terms and conditions set by the service provider may not actual conform with the established policies set by the consumers organisation. Non-compliance with such policies could lead to a loss of reputation and credibility.

Service providers will also adapt their terms and conditions over time and often not inform users of these changes in an explicit manner. This presents the user with a dilemma: *Either they accept terms that are not absolute or a privacy policy that they disagree with, or they do not use the service.* Users may be unaware that the terms and conditions have changed and will use a service governed by policies and agreements which were not the ones originally agreed upon. Even more so, the user may be subject to peer pressure in which the service is also used by the users' peer group and used actively for collaboration. Leaving the service may have an adverse affect upon the users interaction with their peers. Further problems arise when the users themselves fail to adhere to the conditions set out in the agreement e.g. failure to pay or the uploading of inappropriate material. What happens to the users data when this occurs? Will the data be retained and can it be extracted from the service provider? Furthermore a CSP will also change, over time, the terms and conditions attached to the service to further the CSPs own business interests [Bar10]. Users are not in full control over the security of their data and that the protection offered by the service provider is not absolute.

### 3.9.2 Expectations of Privacy

The *Fourth Amendment* of the *United States Constitution* states:

> *"The right of the people to be secure in their persons, houses, papers, and effects, against unreasonable searches and seizures, shall not be violated..."*

Similarly, Article 8 of the *European Convention of Human Rights* states:

> *"Everyone has the right to respect for his private and family life, his home and his correspondence"*

This has resulted in a 'reasonable expectation of privacy' existing, and being guaranteed, over ones home and its contents. Couillard [Cou09] discusses this problem under the auspices of United States Common Law. In the United States the contents of ones briefcase and also school satchel is governed under this expectation of privacy. With Cloud Computing society has naturally extended this reasonable expectation of privacy to cloud services that offer similar functionality e.g. Dropbox. Such services are often described using the terminology *Virtual Containers.* This raises the legal problem of can data stored in the cloud afford the same level of protection as data stored within ones home. The *Katz Test*, used to determine ones reasonable expectation of privacy requires that: a) there was a subjective expectation of privacy over the object; and b) the expectation was reasonable. Implying that there is some form of concealment i.e. opacity, to the object in question. Within the Cloud this would imply that simple password protection or data encryption would be sufficient. Couillard comments that these measures have only been upheld within case law and not in written law. Furthermore, under US Law this expectation of privacy is diminished if the data is handed over to a third-party. The third-party doctrine is that transactional data, data that contains information regarding the transaction itself, can no longer be afforded the same expectation of privacy as the contents of the transaction. The third-party needs transactional data to operate. Couillard contends that a URL can be seen as transactional data and that unlisted links i.e. dynamic ones, do not fall under the fourth amendment. This implies that the contents of a URL that may include authentication tokens and other identifying information will also not be protected by the fourth amendment.

### 3.9.3 National Legislation

Though there is some debate concerning expectations of privacy, existing legislation exists that CSPs should comply with. Such legislation governs the country that the CSP is operating within. Failure to comply with local legislation, or if the information of service users were to be lost, stolen or exposed then such acts could lead have potential legal ramifications for the service provider. Which in turn could result in loss of credibility, loss of reputation and possible reduction in user base [Pea09].

### 3.9.4 Supra-National Legislation

Service provision is not in principle bound at a national level. Users from one country commonly access services located in another. This presents service providers and consumers with problems of under whose laws must the service abide by, and to what degree. Are the consumers allowed to export their data to a foreign country? And what provisions are there for the safety of the data? Amazon.com, for example, offer their *Simple Storage Service* from US (Northern California), European (Ireland) and Asian (Singapore) based data centers to be able to offer services within the different regulatory frameworks of those regions. Furthermore the European Commission's Directive on Data Protection prohibits the transfer of personal data to non-EU nations.

The *Safe Harbour Framework* is a supra-national framework for US based companies to comply with the data protection laws of Switzerland and the European Union [SafeHarbour]. Thus providing a 'safe harbour' for Swiss and EU citizens data residing within the United States. For a US company to comply it must adhere to seven principles, derived from the Directive on Data Protection, of: Notice, Choice, Onward Transfer, Access, Security, Data Integrity and Enforcement. However, there has been significant criticism of this framework concerning the compliance and its enforcement. It was noted in an external study [Con08] that false claims were made by US companies concerning membership and certification. For more information concerning the criticisms made towards the Safe Harbour Framework see Connolly [Con08]; [SEC-2002-196]; [SEC-2004-1323].

## 3.10 Summary

While there are many security issues to be found within the Cloud, one of the common, and underlying, issues is related to the privacy of data stored by the consumer. Such data is susceptible to unwanted exposure, the manner and means of which are dependent upon level of service offered and the protection given. Data stored within the Cloud is more vulnerable and open to attacks than before; its protection is paramount and users need to regain control over the protection of their data from Cloud Service Provider.

Chapter 4

# Cloud Threat Models

*The origin of threats towards data within the cloud are described together with two threat models based upon said lifecycle. The first model represents a user-centric view, the other a Cloud Service Provider point of view.*

## 4.1 Overview

By considering the Cloud as a *remote storage system* i.e. NAS, one can take existing threat models (c.f Hasan, Myagmar et al. [HM+05]) that look towards the area of remote storage and adapt them, as necessary, for the Cloud. Hasan, Myagmar et al. [HM+05] provides a discussion of two threat models for remote storage systems. The first, classifies threats based upon their effect upon the four 'classical' security requirements of confidentiality, integrity, authorisation and availability. The second classifies threats according to how the threats affect data during its lifecycle.

The *CIAA* Threat Model is based upon the four *classical* security requirements of: Confidentiality, Integrity, Availability and Authentication—cf. Chapter 5. Proposed threats are classified according to their relation to each requirement. This model is rather *coarse-grained* with respect to the classification of threats. Threats are categorised upon the effect they have on each requirement and not towards the context of their implementation. A more granular model is required that considers where the threats originate and when the threat is likely to occur during service operation. A threat model based upon the data lifecycle allows for such a model to be constructed.

Remote storage systems can be viewed as a single system by consumers. The presentation of a unified interface through which the consumers interact can also be used to hide the complexity of the underlying infrastructure. This is an important concept and is also shared by Cloud Computing. Cloud Users can also view the Cloud as a 'single' system, however, this viewpoint prohibits for a complete threat model to be established for data in the Cloud. It only provides the users with a view of the flow of their data: *into the cloud and back from the cloud.*

Using these concepts two different yet related threat models for the cloud can be constructed. Section 4.2 provides an overview of the threats in terms of origin, goal, and means. The data life-cycle is enumerated in Section 4.3. Finally, Section 4.4 presents two threat models derived from an overall perspective and from a service user perspective.

**Note.** *The purpose of a threat model is to classify the different threats and vulnerabilities into groups so that they can be resolved accordingly. As Chapter 3 has already discussed possible attacks to data within the cloud, if one were to reiterate these attacks there there will be a duplication of attack descriptions. For brevity the possible threats are not described in detail, furthermore, the attacks listed are not exhaustive.*

## 4.2 Threat Overview

Before the data life cycle threat models are introduced, some time will be spent detailing the origin and nature of the threats.

### 4.2.1 Origin

The origin of threats to data can be divided into the following categories:

**Outsiders** Outsiders are entities that exist outside of the system and attempt to subvert/circumvent the security infrastructure of the service or masquerade as a legitimate service to ensnare users. Their motivation will stem from simple curiosity or from malice.

**Insiders** More serious threats originate from current or past employees of the CSP. Employees will have intricate knowledge of the actual infrastructure including that of security and as part of their remit may have had direct access to the data itself or through other means. Similar to insiders their motive may be out of curiosity or of malice.

**Natural** Although both insiders and outsiders can induce 'errors' within the infrastructure, other errors can occur naturally from the software itself or from hardware failure. For example, when Google pushed a software update to Google Docs [Vas09]. The software malfunction changed the sharing settings of several users documents to include those with whom the affected users had share documents with before.

### 4.2.2 Goals

*Attackers* will also have a goal that drives them. This normally implies targeting a particular asset. These are resources that are either tangible or abstract respectively data and data consistency. Hasan, Myagmar et al. [HM+05] provides an incomplete list of what these assets maybe. One can add to this list more cloud specific assets. An incomplete list of possibly targeted assets includes:

- Communication channel
- Storage media
- Data management software
- Data availability
- Data secrecy
- Data integrity
- Data consistency
- Virtual image availability
- Virtual image secrecy
- Virtual image integrity
- Virtual Image consistency
- Service availability

### 4.2.3 Means

Attackers will usually accomplish their goals by exploiting some technical exploit to gain access to the assets. This can include service hijacking, service impersonation or through poorly secured APIs. If the attacker is a malicious insider they may not need to exploit technical insecurities and will have direct access to the data or will gain access through privilege escalation.

## 4.3 The Lifecycle of Data

The typical lifecycle of data can be describe as the following stages:

**Stage** 1: *Data Creation/Transmission*—this is the initial stage, data is created by the user and then pushed to the Cloud for consumption.

**Stage** 2: *Data Reception*—data is received in the Cloud before being written to storage and logs taken of activity.

**Stage** 3: *Output Preparation*—data is prepared to be returned to the consumer, this involves any transformations that needs to be performed on the data prior to its return i.e. serialisation.

**Stage** 4: *Data Retrieval*—data is received by the consumer from the cloud and has now within the domain of the user.

**Stage** 5: *Data Backup*—the CSP will replicate data for archival purposes. This may involve the transferral of a copy of the data to an external store.

**Stage** 6: *Data Deletion*—data is permanently deleted from the cloud.

## 4.4 Data Lifecycle Threat Models

While the *CIAA* Model can be used to model threats to data, it lacks context. The *Data Lifecycle* Threat Model seeks to classify the threats, first by their placement within the data lifecycle and then using the CIAA model. It is through this classification that one is able to provide some context to the threats and able to produce a more fine grained classification. Hasan, Myagmar et al. [HM+05] introduced a data lifecycle model to describe the threats associated with remote storage. This threat model can be used as a basis upon which a model for cloud resident data can be introduced.

### 4.4.1 Overall Model

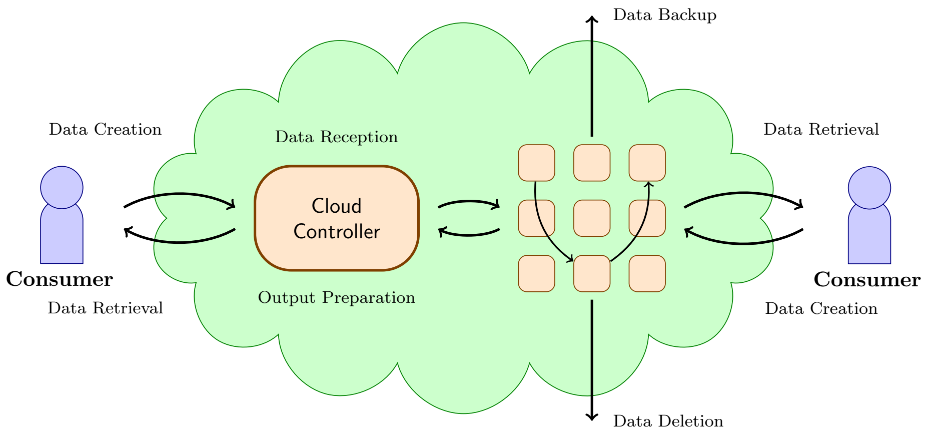


Figure 4.1: The Data Lifecycle Threat Model for the Cloud.

Recall from earlier that in the remote storage context one can assume that the data will be centrally stored within a single store. Within the cloud this cannot be guaranteed. Consumers interacting with both the Cloud Controller and also with the virtualised resources will also be presented with the appearance of a single entity. The data (including the virtual machine) may also be replicated on various virtualised resources and also among different physical machines. By combining this data center model with that of the data lifecycle model for remote storage systems a threat model for the cloud can be produced. Figure 4.1 summarises this new model. The difference between this model and the existing one is that the single data store has been replace by virtual resources. This increases the number of places within the model where these stages can exist. The data creation and retrieval will always occur with the consumer. However, the data reception and output preparation stages can now occur either in the Cloud Controller, if interacting with the controller, or with one of the many virtual resources that will exist. Moreover, as a result one must add an additional stage, representing the internal migration of data within the cloud, to the data lifecycle:

**Stage** 7: *Data Migration*—data is migrated from a resource inside the cloud to another for availability or scalability purposes.

The threats occurring at this new stage can be viewed as an amalgamation of the first and fourth stages of the data lifecycle. That is the sending agent will transmit data and the receiving agent will receive the data. Furthermore, data migration can also be seen as a type of data backup and the threats to those stages can be included as well.

### 4.4.2 User Centric Model

The overall threat model represents an omniscient narrator's view of the cloud; they see and are privy to all aspects. However, one must also consider the user's point of view. In Chapter 3 it was established that the cloud, to a user, represents an unknown risk profile. They are not aware of all aspects of the internal workings of the cloud. Thus, one can also present a user centric

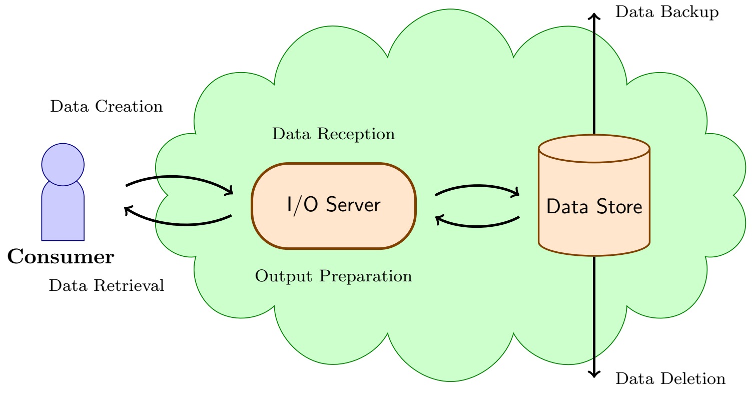


Figure 4.2: The Data Lifecycle Threat Model for Service Users.

threat model that uses as a basis the clients viewpoint. In this model the client views the cloud as a single entity. The more astute reader will notice that this is reminiscent of the original data lifecycle model. Data goes in the cloud, data comes out the cloud, data gets archived, data is deleted. The user is not aware *per se* that the data could migrate between nodes within the cloud. Figure 4.2 illustrates this user centric model.

Chapter 5

# Data Security Requirements

*Grid Computing is used as a basis for defining a set of security requirements for data in the cloud. The responsibility over providing assurances towards these requirements is also discussed.*

## 5.1 Overview

Chapter 4 provided two threat models that one can use to look at the security of data in the Cloud. The threat models presented classified threats according to their relevance within the data lifecycle. This chapter looks to establish a set of security requirements for data held within the Cloud.

*Grid Computing* is a related paradigm to cloud computing, both seek to make available large quantities of computing resources and use them for the processing of large amounts of data [Fos01]. The security issues discussed in Chapter 3 can be used together with cloud computing's similarity with grid computing to enumerate a list of data security requirements based upon existing grid computing requirements [BM03; NR05]. Specifically, Broadfoot and Martin [BM03] provides a detailed set of security requirements for the grid. These are used as the basis for a possible set of, though not exhaustive, security requirements that a solution for securing data in the cloud must look towards. These security requirements will aide when developing or analysing any solution, and also when describing the security offered.

Moreover, it was highlighted that when describing the possible threats to data in the cloud two viewpoints must be taken into account: a) the users; and b) the CSP. This arose as data is either in the cloud or not in the cloud. Thus the security requirements will be the responsibility of: a) the user; b) the CSP; or c) of both the user and the CSP. This shall be addressed for each security requirement presented.

## 5.2 Confidentiality

The data that is to be entrusted to the cloud may be of a sensitive nature and will thus be subject to several confidentiality measures. Although confidentiality of data is primarily seen as being solved via encryption, as discussed in Broadfoot and Martin [BM03], other aspects need to be considered.

**User and Resource Privacy** Within the Cloud, confidentiality of data also extends to how the data is being processed/used and also the users actions. The means by which CSP can store or process the data is bound by law and these laws must be adhered to. Such data includes and is not limited to: auditing records indicating access attempts and changes (and their results) to the data; properties of the data including size, access policies and origin; and even the existence of the data itself.

**Deducible Data** Recall from the previous Chapter in Section 3.6.2 that hidden information pertaining to an individual can be deduced from existing information i.e. Bob's sexuality. An attacker should not be able to use existing information or information relating to the confidential data i.e. meta-data to deduce any other information. Such attacks should be made as difficult as possible. For those who have multiple personae on the web relating to the different facets of their private life. The ability of an attacker to link the two should be hard.

The requirement of confidentiality is an aspect that both the user and CSP need to be made aware of. Specifically, the confidentiality of the plain-text data itself should be the responsibility of the user before it goes into the cloud. Guarantees towards *user and resource privacy*, and deducible data is best made by the CSP. They are in a better position to provide such guarantees.

## 5.3 Remote Access

The Cloud, by nature, is inherently a 'public place'. Services are exposed over HTTP, a public medium. Access to these services need to be controlled and access kept to authorised personnel. Moreover as the data is held remotely, trust needs to be established with the service and with the security provided by the service over the data itself. Access to the data needs to be regulated. CSPs must ensure that entities trying to access the data are not only who they say they are (authentication) but also that they have the right to do so (authorisation) This is made more difficult as CSP will be interacting with multiple users from multiple companies (domains) each of whom will require different management and access policies; and all done remotely.

**Authentication** CSPs must ensure that those trying to access the service are who they say they are. Unauthenticated users and impostors should not be able to access the data. The identity of the entities must be assured. This will imply some form of identity management.

**Authorisation** Once the identity of an entity has been established access to the data held by the CSP needs to be regulated and controlled. Authenticated entities should not be able to access data that they are not authorised to access. For example, two users from different companies should not be able to access each others remote data held by the CSP unless the access has been explicitly allowed.

**Location** Users may be accessing the service/resource from different locations. Authentication of the user should always be performed and should not be linked to the device from which the entity accesses the service.

**Revocation** An important requirement is that of revocation. The revocation of access to individual data and to the service itself must be permissible.

With remote access, guarantees towards location privacy, authentication of identity and authorisation to the data that is resident within the cloud should be made by the CSP. Revocation, and thus assignation, of access to the data should be made by the user themselves.

## 5.4 Non-Repudiation

Both the CSP and user should not be able to deny the origin or refute the integrity of data. Moreover, a verifiable record of the data's lifecycle should exist. The lifecycle of the data and the operations performed on the data should be attestable if a CSP attempts to defraud a user and *vice versa*. This is especially important if flexible payment models are used. A dishonest CSP could claim that more resources were used by the user and thus be in a position to bill the consumer more than what was actually the case. This implies that records need to be kept concerning usage that can be verified by both users and consumers. Guarantees towards non-repudiation should be made by both the user and the CSP.

## 5.5 Integrity and Consistency

The mobility of data within the cloud only increases the threats that can affect the integrity of data. Data is being transported to-and-from the user and service providers, and also internally within the cloud. The integrity of the data must be guaranteed when it has been placed within the Cloud. Consistency problems can arise from omission and commission failures. Omission failures occur when an entity fails to act upon input. Commission failures are those that occur when an entity though responds to input the output is not what was expected. It is possible for hardware to fail or connections to be lost at which time the data may be in an inconsistent state or unrecoverable. If data is replicated for some reason e.g. to combat availability, scalability or archival purposes, the consistency of the replicated data must be ensured. The consistency of replicated data can be affected by omission and commission failures. Also, consistency problems can arise during multi-author collaboration when access to the data is performed concurrently.

With regards to integrity and consistency this is a joint responsibility, though it can be divided cleanly between the two entities. The user can make guarantees towards the integrity of the data before it goes into the cloud i.e. pre-cloud insertion integrity, and the CSP needs to ensure the integrity *and consistency* of the data once it is in the cloud i.e. post-insertion.

## 5.6 Availability and Fault Tolerance

Another problem with entrusting data to a service provider is ensuring the availability of the data once it has been placed within the cloud i.e. resource availability. This is essentially a guarantee that can only be made by the CSP themselves. Users only have the assurances made by their service provider that data will be made available. If the data were to be made unavailable for some reason, users will not be able to access their data and become inconvenienced. The inconvenience caused could also lead to profit loss for both the user and the CSP. Moreover, internally the nodes within the Cloud must also be resilient to node failure and the data held on the nodes must still be available. Internally the Cloud must be fault tolerant.

**Note.** *Availability also overlaps with remote access requirements. Access rights to the data can also be seen as a form of availability, however this was already discussed in terms of authorisation in Section 5.3.*

## 5.7 Summary

Below a summation of the requirements according to their division is given.

| **User** | **Joint** | **CSP** |
|---|---|---|
| Data Confidentiality;<br>Access Right Revocation;<br>Access Right Assignment;<br>Pre-Cloud Insertion Integrity | Non-repudiation | User and Resource Privacy;<br>Deducible Data;<br>User Authentication;<br>Location Privacy;<br>Post-Cloud Insertion Integrity and Consistency;<br>Resource Availability;<br>Fault Tolerance |

Chapter 6

# The Trappings of a Solution

*A discussion towards the security of, and how best to protect data in the Cloud is provided. The notion of data privacy is addressed. The degree of trust afforded to a Cloud Service Provider is analysed. Finally, responsibilities over guarantees towards security requirements are discussed.*

## 6.1 Overview

Over the preceding chapters the notion of data security within the cloud was discussed. One of the prevailing factors has been the unwanted exposure of data be it a result of a software malfunction or malicious CSP. When entrusting data to the cloud the data creators i.e. service users and CSPs, need assurances over access to their data. In essence data creators need to regain control over this access, data creators need to become empowered. This chapter discusses what potential solutions should aim towards when looking to *empower* those whom create data. More precisely, the discussion begins with a look at what this empowerment should entail (Section 6.2) before discussing the trust that can be afforded towards a CSP—Section 6.3. The responsibilities concerning the protection of data is discussed within Section 6.4. Concluding this chapter is Section 6.5 in which the salient points over what makes a solution are presented.

## 6.2 Defining Empowerment

Within computer security the protection of one's data can be seen as being synonymous with the protection of one's privacy. The 'right to privacy' is a fundamental right and is enshrined in many a countries constitution. A user empowered over the privacy of their data can be seen as being empowered over the protection of their data. A thorough grasp of this *notion* of *data privacy* is fundamental when building a solution to empower users. As such a better solution can thus be designed and what it means for a user to become empowered can be determined.

However one of the major problems with this interpretation of data privacy, is its definition. Data Privacy is a rather vague and often misunderstood term. For example, take three existing solutions: None of Your Business (NOYB) [GTF08]; *Privacy Manager* [MP09]; and Content Cloaking (CoClo) [DVZ10]. Each of these three solutions each have a different take on how to protect the privacy of data.

- *Encryption.* The CoClo solution dictated the encryption of data prior to its insertion into the cloud. Data was hidden completely from unauthorised users and the CSP.
- *Obfuscation.* With *Privacy Manager* data was obfuscated. While 'obfuscation' does not necessarily imply the encryption of data, obfuscated data can still nonetheless be operated upon by a CSP with the CSP not learning anything about the underlying data. Examples of obfuscation techniques can be found in Kantarcioglu [Kan08].
- *Contextual Integrity.* The NOYB solution sought to destroy the link between the data and its creator, as well as hide the data itself.

The remainder of this section will discuss each of these takes on data privacy and will discuss how a model based upon Kafka's *The Trial* best describes how users can become empowered within the Cloud.

### 6.2.1 Obfuscating Data: 1984 was not a good year

The encryption and obfuscation of data to protect privacy is by far the most common response when seeking to protect the privacy of data. All the aforementioned solutions use this technique to some degree. Obfuscation techniques are a dogmatic response to the fears envisaged by the populistic metaphor of the *Orwellian 'nightmare'*. Originating from George Orwell's famous novel *1984*, this metaphor's core theme is the harm that arises from the constant surveillance of individuals. Through the constant surveillance of an individual 'Big Brother' i.e. the government, is able to use this information for both useful and nefarious purposes. In the novel the harm depicted relates to how the information was used to curb and control social activities. Naturally, this Orwellian 'nightmare' has led to the view that the data collector, in this setting the CSP, is inherently malicious, seeks to cause harm and should not at all be trusted. As mentioned previously the initial response to alley these fears is to prevent the collection of data through obfuscation techniques including that of data encryption: The CSP is not trusted at all. This was noted in Chapter 3 with the unknown risk profile presented by the CSP which in turn resulted in the user-centric threat model presented in Chapter 4. In this threat model consumers cannot make any guarantees, themselves, towards the data once it is in the cloud.

However, when using obfuscation based techniques if the CSP is not aware of the obfuscation it will not necessarily be able to correctly process the information: *some functionality of the service will be lost* [MP09]. This loss of functionality may also have a detrimental effect upon the service's *user experience*. In D'Angelo, Vitali and Zacchirolo [DVZ10] the authors note that by using client side encryption all server side functionality such as document search will be lost. This 'loss of functionality' represents a lack of trust between the service user and CSP. A different approach to defining privacy and what the protection of data entails is required. Such a solution needs to bridge the gap over the control of the privacy of the data and the functionality offered by the CSP.

### 6.2.2 Contextual Privacy

The NOYB solution uses a privacy model based upon the contextual integrity of data: *Contextual Privacy*. Originating from Nissenbaum [Nis04], the privacy of data is based upon its context. Data Privacy holds if the data cannot be linked back to the originator of the data: *the data will be viewed out of context.* 'Big Brother' will be able to collect data and use the data but will not be able to link the data back to the data originator. Ostensibly, this appears to be madness, one's data will still be public. However, the success of the contextual privacy is dependent on what the 'context' of the data is and also what the data itself is. With NOYB the original context of the users data is their Facebook profile [GTF08]. This information is then broken down into atoms representing data items that are common to all i.e. christian name, age and gender, before being randomly distributed among other profiles. While, the person's information will still reside on Facebook it will be viewed out of context of the persons own profile. Facebook, will thus be able to use the information but will no longer be able to link the data back to the original users. Furthermore, the non-common data items such as email addresses, telephone numbers and addresses are first broken down into atoms representing individual characters. While this is an interesting privacy model, users' data is nonetheless publicly available.

### 6.2.3 A Kafkaesque Approach

Solove [Sol07] argues that a better understanding of privacy can come from not seeing privacy as a single concept but rather as a pluralistic one. The privacy of data will take on various forms depending upon the context under which the data is being examined. This resembles the views expressed in the previous section in which the privacy of data holds if the data stays 'out-of-context' i.e. cannot be linked back to the data originator. The underlying metaphor used by Solove [Sol07] stems from Franz Kafka's *The Trial*; the approach is *Kafkaesque*. *The Trial* depicts a bureaucracy that uses people's information to make decisions about them yet prevents the peoples ability to participate in how their information is used. Much like the problems associated with users placing their data within the cloud. Here the collection of data is not the main issue, rather it is the unauthorised and unchecked processing i.e. storage, use and analysis, of the collected data.

### Taxonomy of Privacy

By conceptualising privacy as what happens to data once it has been collected Solove [Sol07] demonstrates that a taxonomy of privacy violations can be constructed. The taxonomy presented by Solove presents four general categories of privacy problems. The four general categories are: a) *Information Collection*—representing the collection of information itself; b) *Information Processing*—representing the use of information by the data collector; c) *Information Dissemination*—representing the unauthorised release of information; and d) *Invasion*—representing the use of information to intrude upon an individuals life.

Under the Orwellian model there is only 'one' category of privacy violation, that of information collection. One can also view the taxonomy by Solove as relating to the potential lifecycle of data in terms of privacy violations. First information is collected, it is then processed, disseminated and then used purposefully to violate ones privacy. With the typical response to privacy woes being preventing the collection of data i.e. response to the Orwellian model, this response effectively curtails *all* privacy violations. However, the prevention of the collection of data is a greedy solution, it seeks to solve all problems in one fell swoop. This is not the most effective solution, it implies loss of service functionality when using obfuscated data—cf. Section 6.2.1. With this taxonomy a new and more informed response can be constructed.

### The Kafkaesque Response

The use of this taxonomy allows for the shift of the discussion concerning privacy woes away from viewing the CSP as inherently malicious i.e. it wants to collect ones data for nefarious purposes. The focus is now on the data collector as an entity that (mis)uses one's data regardless of the user's own wish. This is the comment made in *The Trial*. The collector need not be an overtly malevolent or benevolent entity, their character is not the main issue. The issue is the loss of control that the user has over the use of their data; they have been left powerless.

The objective is to now empower the users over the use of their data. This is a better fit to the privacy woes presented by the Cloud than that of the default anti-collection response. The emphasis is letting the users take control over how their data can be used by the CSP. The empowerment stems from giving the users the *choice* of what happens to their data and preventing the CSP from making such choices on their behalf. However, this new approach implies that some trust needs to be placed with the CSP, to respect the users *choice*. In the next section this trust, between users and Cloud Service Providers, is discussed.

**Remark.** *The important notion here is that of* choice*, the user should be able to choose what, if any, information should be collected and then used.*

## 6.3 In CSP, we Trust?

Trust is an important notion. To trust an entity implies that one has confidence or faith in that entity. In the previous section the trust imparted to the data collector/CSP by the user varied and was dependent upon the view the user had of the CSP. For instance, the Orwellian approach stipulated no trust in the data collector, while the Kafkaesque approach stipulated that one could trust the CSP somewhat. From these differences a 'taxonomy of trust' can be constructed that describes how a user views and comprehends the actions of the CSP. They are:

- **No Trust**—the Orwellian approach, one does not trust the CSP at all. Used by the CoClo solution [DVZ10].
- **Some Trust**—the users trust the CSP to some degree. Used by the NOYB and *Privacy Manager* solutions [GTF08; MP09].
- **Complete Trust**—the user completely trusts the CSP.

The remainder of this section will discuss each of these modes.

### 6.3.1 No Trust

With the Orwellian approach, one thought of the CSP as being inherently malicious: *In CSP they did not trust.* The typical response was to interact with the CSP using obfuscated data. As was discussed previously, this approach has one major drawback. When one starts to not trust the CSP with data and thus send obfuscated data, some functionality of the service will be lost if the CSP is unprepared, or such measures cannot fit into operation of the service [MP09]. This is especially pertinent if one sends encrypted data. Having no trust in the CSP appears to lead to more problems than it does solutions. Moreover, from a legal point of view the transmission of obfuscated data may result in a breach of any service level agreement that the data creator has signed, or agreed upon, with their CSP.

### 6.3.2 Complete Trust

At the other end of the spectrum from having no trust in the CSP is to have complete trust. While this was not specified by any of the solutions mentioned thus far, its use has been seen. *Grendel* is a complete service-side solution touted by Hedlund [Hed10]. Grendel stores users data in an encrypted format and only decrypts the data when it is being used by an authorised entity. This approach implies that the user has complete faith in the CSP to provide security over their data. The user has effectively acquiesced to the CSP providing all guarantees over the protection of their data. This acquiescence also extends to how the data is used. Users are left with no real empowerment over the control over the protection of their data. Therein lies a problem, all the security is now service-side. Any consumer must have complete trust in the CSP to keep their data secure. Though the cryptographic operations maybe secure the CSP still is responsible entirely for the security of the data. Moreover, key management and storage is also completely service-side, the client must have faith in the CSP concerning key management and storage. This leads one to acknowledge that there is a trade-off between trusting, and thus letting, the CSP protect and use data, and its security.

**Note.** *Grendel's operation for securing and sharing data is the same as that of CoClo. Unlike CoClo, however, all the operations are performed service-side, thus relieving the client of the burden concerning the security of their data.*

### 6.3.3 Some Trust

The third and final mode of trust implies that the user has some trust in the operation of the CSP but is allowed to have reservations about said operation. Such trust can originate from the service level agreements provided by the CSP (see Section 3.9.1) in which a certain standard of protection has been promised. The reservations, will originate from the unknown risk profile of the service—see Section 3.8. With this mode of trust, one needs techniques that allow for the user to trust the CSP to some degree over the protection of their data and remain empowered. Two such examples that have already been mentioned in this chapter were NOYB and *Privacy Manager*. They allowed the user to use the service and more importantly not lose functionality provided by said service. However, with these examples, particularly that of *Privacy Manager*, it was noted that for the solution to work the CSP needs to have sufficient knowledge of the techniques used by the user to protect their data so that the CSP can work with the data and respect the users choice.

## 6.4 Security Responsibilities

When protecting data that is to be placed in the Cloud, one also has to think about who is responsible for the protection of data. That is for which security requirements are users and the CSP liable. Throughout the discussion presented in this chapter a recurrent theme is that of both the user and the CSP playing a role. For instance, Section 6.2 mentioned that the Kafkaesque approach implied both the user and the CSP needing to work together for privacy to hold. This was reiterated within Section 6.3 with the *some trust* view. From this, it can be argued that both the CSP and user need to take active responsibility over this protection. Such as view was stated within Chapter 5 and a division of responsibility was summarised in Section 5.7. The service user is responsible for the confidentiality and integrity of data prior to its insertion into the cloud, and for specifying access to the data post insertion. The CSP is responsible for the security guarantees that the users themselves cannot make such as: data availability, accessibility (as specified by the user), and the consistency and confidentiality of the data once it has been placed within the cloud.

## 6.5 Towards a Solution

When designing a solution that allows those that place their data in the cloud, one needs to take into account the following salient points.

- *User empowerment does not alone equal Data Obfuscation.*

When ensuring the confidentiality over one's data the aim should be for the data creator to gain control over how the data is being used rather than solely preventing its initial collection. This will include the data creator being in a position to dictate to whom access to their data should be granted.

- *The user need not be responsible for ensuring all security guarantees.*

By design the service user has entrusted their data to some service. It is obvious that some form of co-operation must occur between the service user and the CSP over the protection of this data. A service user cannot be responsible for all aspects governing the confidentiality of their data. A good solution must recognise this and allow the service user to have trust in the CSP's ability to offer data protection.

- *Some CSPs may be evil but some are more evil than others.*

The maliciousness of a CSP will differ from CSP to CSP. The degree of trust afforded to each individual CSP will as a result also differ. Service users who can recognise this difference should have the option of allowing trusted CSPs access to their data, and also be able to revoke such access.

# Part II

# Predicate Based Encryption

CHAPTER 7

# Introduction to Predicate Based Encryption

*A high-level overview of Predicate Based Encryption schemes is given looking at their: definition, characteristics, access control, and cryptographic key composition.*

## 7.1 Overview

Predicate Based Encryption (PBE), represents a family of asymmetric encryption schemes that allows for selective fine-grained access control as part of the underlying cryptographic operation [KSW08]. The origins of PBE are in Identity Based Encryption (IBE) [Sha85]. In IBE schemes an entity's encryption key is derived from a simple string that represents the entity's own public identity e.g. an email address. For example, given an entity Albert his corresponding encryption key will be $\mathsf{Enc}(Albert)$ == albert@foobar.com. During encryption, the resulting cipher-text will effectively be labelled with the string representing the encryption key, the entity's public identity. An entity's decryption key will be derived from the same string used for the encryption key e.g. Albert's decryption key will be derived from his e-mail address. On recipt of a cipher-text message the recipient will be able to decrypt the cipher-text if and only if the two identities, contained within the decryption key and cipher-text, are 'equal'. PBE schemes offer a richer scheme in which an entity's 'identity' can be constructed from a set of attributes and decryption is associated with access policies that offers a more expressive means with which to describe the relation between the attributes.

Generally speaking, within PBE schemes entities and cipher-texts are each associated with a set of attributes. These attributes are used to describe some aspect of the entity, the data that is being encrypted, and the environment. An entity will be able to decrypt a cipher-text only if there is a *match* between the attributes of the cipher-text and the decrypting entity. Matching is achieved through predicates (access policies) that denote: a) the set(s) of authorised attributes that an entity must possess in order to decrypt and access the plain-text; and b) the relationship between the attributes. Taking a dating website as an example. To indicate that only females over 5′9″ with an ability to speak Dutch or British English will be able to access encrypted data a user can use the following policy:

$$\text{gender:female} \land (\text{speaks:dutch} \lor \text{speaks:british-english}) \land \text{height} >= 5'9''$$

Various constructions of PBE schemes have been proposed that use general predicates (represented as boolean formula) or specific predicates such as *equality*, *hidden vector* or *inner product.* The choice of predicate will have a direct affect upon the scheme, its characteristics and the composition of the access policies. Moreover, the placement of the predicate (either with the cipher-text or the decryption key) has a great affect upon the workings of a PBE scheme. The affect of predicate placement and other characteristics are discussed in Section 7.3. This chapter provides a high-level overview of PBE schemes. Details concerning the construction of PBE schemes are to be found in Chapter 8. Their use as part of a cryptographic system is discussed within in Chapter 9.

## 7.2 Definition

Common to all PBE schemes are four operations allowing for encryption, decryption and key generation. The precise value for encryption and decryption keys is dependent upon both the construction of the scheme and placement of predicates. This is defined later in Section 7.3.2. A general PBE scheme is defined as follows:

**Definition 1** (General Predicate Based Encryption (PBE) scheme)**.** *A general PBE scheme consists of the four operations:*

- ***Setup**: initialises the crypto-scheme and generates a master secret key* MSK*, used to generate decryption keys, and a set of public parameters* MPK.

$$(\mathsf{MSK}, \mathsf{MPK}) := \mathsf{Setup}() \tag{7.1}$$

- ***KeyGen**: generates a decryption key* Dec(*entity*) *based upon the master secret key and some entity supplied input.*

$$\mathsf{Dec}(entity) := \mathsf{KeyGen}(\mathsf{MSK}, input) \tag{7.2}$$

- ***Encrypt**: encrypts a plain-text message $M$ using the public parameters and supplied encryption key for an entity.*

$$CT := \mathsf{Encrypt}(M, \mathsf{MPK}, \mathsf{Enc}(entity)) \tag{7.3}$$

- ***Decrypt**: decrypts a cipher-text if and only if the attributes held by the entity can satisfy the access policy.*

$$M := \mathsf{Decrypt}(CT, \mathsf{MPK}, \mathsf{Dec}(entity)) \tag{7.4}$$

**Note.** *Several constructions also provide a fifth operation: **Delegate** in which an entity will delegate decryption to a secondary entity. This operation is not common to many a PBE construction and for brevity it shall be omitted. The more interested reader should consult Bethencourt, Sahai and Waters [BSW07] for an example of a scheme that supports delegation.*

## 7.3 Characteristics

PBE schemes will each possess different characteristics. Theses characteristics are based upon: a) the type of predicates used—Section 7.3.1; b) the placement of the predicate—Section 7.3.2; and c) visibility of attributes and predicates—Section 7.3.3. These characteristics have been derived and collated from several existing sources governing PBE schemes. Primarily Katz, Sahai and Waters [KSW08], Goyal, Sahai et al. [GS+06], Waters [Wat10] and Bethencourt, Sahai and Waters [BSW07] were consulted. PBE schemes can also be categorised further according to differences in how the scheme was constructed. The different ways in which PBE schemes can be realised is discussed in Chapter 8.

### 7.3.1 Types of Predicates

When looking at the different PBE schemes, three groupings (or family) of schemes emerge based upon the predicates being used, the scheme's aim and the schemes construction.

**Identity Based** Identity Based Encryption (IBE) schemes are used to encrypt messages using a single attribute that refers to the users identity i.e. email address or passport number Shamir [Sha85]; Boneh and Franklin [BF01]. In these schemes the access policy is also comprised of a single attribute. To add extensibility to IBE schemes these 'single' attributes were extended using string concatenation to encode more information.

**Attribute Based** Attribute Based Encryption (ABE) schemes [GS+06] further generalises upon IBE schemes and uses general predicates styled as boolean formula covering operations such as conjunction, disjunction and thresholds. The attributes themselves need not necessarily refer to an entity's identity, or even to an entity, and can refer to non-identity related aspects such as `TCP/IP` port numbers and addresses.

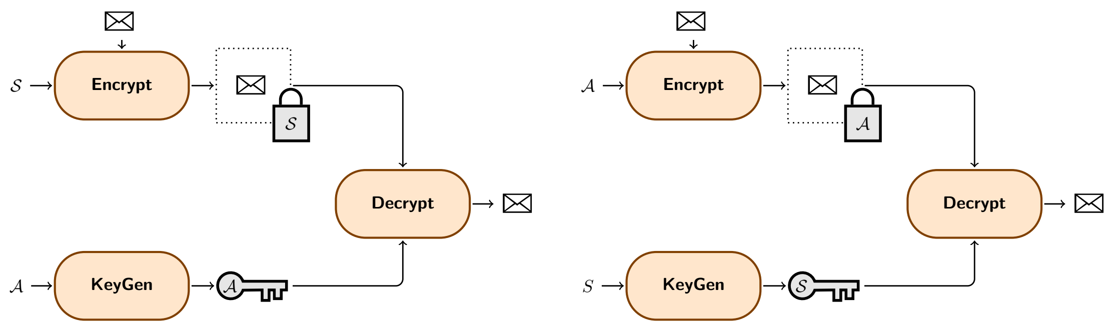
**Specific Predicate Based** The final family of schemes are those that use specific predicates during their construction. Although these schemes can be referred to by the predicate being used, the general term Predicate Based Encryption can also be used. Specific predicates that have been used include that of *inner product* [KSW08] and *hidden vector* [BW07] predicates.

### 7.3.2 Predicate Placement

The placement of the predicate will have an affect upon the values that the cryptographic keys can take and ultimately the parameters used by the operations described at the beginning of this section. Ciphertext-Policy schemes are schemes in which the access policy is associated with the cipher-text. Key-Policy schemes are those in which the access policy is associated with the decryption key. The difference between these two types of scheme, when used as part of a PBE crypto-system, is discussed further in Chapter 9.

**Key-Policy** In Key-Policy (KP) schemes an entity's decryption key is derived from an access policy $\mathcal{A}$, and encryption keys are sets of attributes $\mathcal{S}$. During encryption a set of attributes $\mathcal{S}$ is used to encrypt a message, and $\mathcal{S}$ is also *encoded* alongside the resulting cipher-text for use during decryption. A decryption key will be able to decrypt a cipher-text if the access policy used to generate the decryption key can be satisfied by the attributes encoded alongside the cipher-text. Figure 7.1a illustrates the overall operation of a Key-Policy scheme.

**Remark.** *An alternate way to think of KP schemes is that the decryption key tells the decrypting entity what 'types' of cipher-text they can decrypt. The encryption key, or encryption process, will assign 'types' to the cipher-text.*



(a) Key-Policy Scheme (b) Ciphertext-Policy Scheme

Figure 7.1: Outline of Ciphertext-Policy and Key-Policy Schemes' Operation.

**Ciphertext-Policy** In Ciphertext-Policy (CP) schemes encryption keys are access policies and an entity's decryption key is derived from a set of attributes $\mathcal{S}$. During encryption an access policy $\mathcal{A}$ is used to encrypt messages, and $\mathcal{A}$ is also *encoded* alongside the resulting cipher-text for use during decryption. Decryption of a cipher-text will occur only if the values contained within the decryption key i.e. within $\mathcal{S}$, can satisfy the access policy embedded alongside the cipher-text. Figure 7.1b illustrates the overall operation of a Ciphertext-Policy scheme.

**Remark.** *An alternate way to think of CP schemes is that the encryption key explicitly states the conditions under which decryption can be performed. While the decryption key* is *the condition itself.*

### 7.3.3 Privacy

The visibility of the access policies and attributes will differ between schemes. By design all PBE schemes are Payload Hiding (PH). This ensures that the payload cannot be accessed by a malicious entity. A PBE scheme is Attribute Hiding (AH) if the scheme also hides knowledge of the encryption key i.e. attributes/access policies, used to encrypt the cipher-text. During decryption when an entity attempts to access the plain-text they will only learn if the decryption procedure is successful or not and will not learn anything else concerning the cipher-text and its attributes/access policy. While all schemes by default are payload hiding, attribute hiding schemes are dependent upon the underlying predicates used to realise the scheme. Such functionality is achieved when using *inner product* [KSW08] and *hidden vector* [BW07] predicates.

## 7.4 Access Control

Through the use of attributes and predicates, a more richer form of access control has been built into the cryptography when compared to more traditional asymmetric encryption schemes such as *RSA* and *El Gamal* [RSA78; ElG85]. In traditional asymmetric schemes cipher-texts can only be decrypted using a single key that is paired with a single encryption key. One can summarise the relationship between decryption keys and cipher-texts as being one-to-one: *one cipher-text can be decrypted only by one decryption key.* However, with PBE schemes this relationship is more flexible. Decryption in a PBE scheme will occur if the predicate can be satisfied by a given set of attributes. This relationship can be described as being one-to-many: *one cipher-text can be decrypted by many decryption keys.*

By specifying access control in terms of attributes and predicates, the access control offered by PBE is analogous to Attribute Based Access Control (ABAC) and involves the assignment of descriptive attributes to entities, resources and the environment [YT05]. Access to a resource is decided upon by access policies that described the sets of attributes required, often as a boolean formula, for access to occur. ABAC is considered to be more flexible and scalable when compared to other existing access control techniques such as Role Based Access Control (RBAC) [SC+96] and Lattice Based Access Control (LBAC) [SS94]. Moreover, ABAC is able to combine the functionality of both RBAC and LBAC into a single access control model. The authorisation architecture for ABAC consists of the following four actors:

- An Attribute Authority (AA) is responsible for the creation and management surrounding attributes used to describe the resources, entities and the environment respectively.
- The Policy Authority (PA) is responsible for the creation and management of the access policies that govern access to resources.
- The Policy Decision Point (PDP), is where the access policies governing access to a resource are evaluated against the attributes attributed to the requesting entity, resource and the environment.
- A Policy Enforcement Point (PEP) is a fixed point where the right of the entity requesting access to a resource is challenged. This differs from the PDP in that here the challenge is issued, and at the PDP the challenge is performed.

PBE does not replicate the functionality seen in ABAC completely. Unlike ABAC the environment, entities and resources are not assigned to attributes directly, they are attached to cryptographic keys. The use of which, will have an affect upon the access control. Furthermore, the distribution of these four actors to entities will differ based upon the placement of access policies i.e. the type of PBE scheme. Moreover, there is no central point at which policy enforcement and decision will occur. The remainder of this discussion is deferred until Chapter 9.

## 7.5 Attributes

The set(s) of attributes used to label both entities and cipher-texts, and construct access policies, originates from a universe of attributes $\mathcal{U}$. The size of the attribute universe is dependent upon the construction of PBE being used. The universe of attributes is either: a) **Small**—if the number of attributes is fixed and the attribute values specified as part of the system instantiation; or b) **Large**—if the number of attributes is unlimited in size and values can be specified later.

While the precise semantics governing these attributes are dependent upon the setting of use much can nonetheless be discerned. Attributes are either textual or numerical in value, and can also be represented as key-value pairs in which the key can possess multiple different values. Within PBE attributes are used either as an identifier or as a descriptor of some (in)tangible aspect of either the entity or data that is to be encrypted. Attributes can be divided roughly into three categories corresponding to whom and what they reference:

- *Entity Attributes* Attributes that define the identity and other related characteristics associated with an entity. Such attributes may include an entity's name, pay grade, security clearance level and organisation affiliation.

- *Resource Attributes* Attributes that describe characteristics associated with the resource that is to be encrypted. Taking `TCP/IP` connections as an example; the source and destination IP address and port numbers could be used as attributes within the cryptographic system.

- *Environment Attributes* There will often be attributes that relate neither to a resource nor an entity e.g. the current date and current locale. Such attributes refer to the environment in which both entities and resources reside.

**Remark.** *It will be the case that entities or cipher-texts will have several attributes in common, for example: office number, address, IP-address* et cetera. *It is this commonality that allows for the selective fine-grained access control to exist.*

### 7.5.1 Attribute Uniqueness

Attributes must be uniquely named and their meaning clearly distinguishable from other attributes. Moreover, related types of attributes such as dates should be presented in a common format to ensure their canonicity. This could otherwise give rise to the malicious use of attributes. For example, take the use of attributes to capture the source and destination `IP` addresses within a `TCP` connection. Naively, one would be tempted to simply allow for `IP` addresses to be captured and used in their standard form i.e. `127.0.0.1`. However, this form conveys no information concerning whether or not this is the source or destination address within the `TCP` header. A more suitable and canonical form would be `DEST.ADDRESS=127.0.0.1`. This conveys more information concerning the meaning of the attribute and leaves rise to less confusion concerning as to if this were the destination or source address.

### 7.5.2 Compound Attributes

Attributes with PBE schemes are represented within a single set. When attempting to satisfy an access policy, all possible combinations of the attributes issued can be used. While this is useful for natural singleton attributes, compound attributes representing natural combinations of single attributes are more difficult to specify and use. Such attributes arise when re-enforcing the notion of attribute uniqueness. A typical example of where compound attributes come into existence are when attributes are used to refer to entities. Within an organisational setting individuals are often given some short term responsibility. For example, within a university PhD students are often tasked with teaching assistant duties. These duties will change yearly, if not each semester. This would result in attributes of the form: `TA:<course code>:<year>` being used to identify a teaching assistant for a course, for a particular year. This causes problems as this precludes the use of the attribute in an access policy that grants access to *all teaching assistants* i.e. `TA`. PBE schemes are not designed to decompose compound attributes.

Naively, one may be tempted to solve the problem of compound attributes by stating that when using *only* singleton attributes the 'construction' of compound attributes can be achieved within access policies using conjunctions. For example, `TA:<course code>:<year>` would become: `TA` ∧ `<course code>` ∧ `<year>`. However, this can give rise to the malicious use of attributes to satisfy poorly designed access policies if the attributes were not uniquely named.

Although there have been attempts at the creation of PBE schemes that alleviate the problems associated with compound attributes (see Bobba, Khurana and Prabhakaran [BKP10]) the problem of compound attributes still remain an important issue when using PBE schemes.

## 7.6 Access Policies

Predicates are used to define the required sets of attributes needed, often in terms of boolean formula, by the decrypting entity for decryption to occur. Within PBE the term *Access Policy* can be used interchangeably with the term predicate, and within Key-Policy schemes with decryption key. This section presents an overview of access policies, their definition and limitations.

### 7.6.1 Expressiveness of Access Policies

The expressiveness of an access policy is dependent upon both: a) the predicates used by the scheme; and b) other mathematical constructs used within the scheme's construction. That is, the permissible operations available when constructing an access policy are not universal and will differ. The reasons for this difference, aside from differences in underlying predicates, is discussed further within Section 8.3. This difference, in expressiveness, can be summarised in terms of the operations available and the number of arguments that these operations allow. The most common form of predicate seen supported, is that of a two-argument operator supporting conjunction and disjunction operations. Other forms seen include the use of multi-argument boolean operations supporting threshold, numerical comparisons and attribute negation. Section 7.7 presents several PBE schemes and notes the expressiveness of their access policies.

Though there are constraints over the exact operations permissible, these constraints reflect the internal use and representation of the predicates within the crypto-scheme itself. The internal representation represents a normalised form that the predicates must take. Implying that more complex policies can be constructed as long as they can be transformed into the required normalised form.

### 7.6.2 Definition

The disparity between the expressiveness of access policies in PBE schemes increases the scope of the discussion greatly. For simplicity the scope of the discussion concerning access policies will be restricted to the use of general predicates. General predicates can be represented as a boolean function $\mathcal{F}$. The definition for an access policy is as follows:

**Definition 2** (Access Policy). *Adapted from [BL88, Section 2]: Given a universe of attributes $\mathcal{U}$, an access policy $\mathcal{A}$ is a boolean function $\mathcal{F}$, indexed by a set of attributes $\mathcal{P} = \{P_1, \dots, P_n\}, P_i \in \mathcal{U}$. The function $\mathcal{F}$ supports the following boolean operations:*

- Disjunction*: $\vee$—1 out of $n$ attributes.*
- Conjunction*: $\wedge$—$n$ out of $n$ attributes.*
- Threshold*: $T_n(a_1, \dots, a_m)$—$n$ out of the following $m$ attributes must be present.*

*The operations define the set $A$ of $a \subseteq \mathcal{P}$ for which $\mathcal{F}$ is* true *whenever the variables indexed by a set in $A$ are also set to* true*. That is, the function $\mathcal{F}$ returns true if and only if an entity is able to satisfy at least one $a \in A$.*

**Remark.** *Conjunction and disjunction operations represent specific instances of a threshold function when $n = m$ and $n = 1$ respectively.*

**Note.** *With respect to multi-valued attributes standard set notation can be introduced as a short hand in certain cases. When the same attribute is referred to in a series of disjunction operations the following can be specified att $\in \{a_1, \dots, a_m\}$. Where 'att' refers to the key and each $a_i$ is the different values. Similarly when assigning an attribute set that allows for multiple values to be assigned the following can be specified: att $= \{a_1, \dots, a_m\}$.*

### 7.6.3 Additional Operations

Although Definition 2 provides a healthy set of operations that can be performed upon attributes using general predicates, other operations can also be used.

#### Numerical Comparison

Numerical operations allow for the existence of key value pairs to be specified, and used within access policies to construct range queries. In supporting PBE schemes the standard set of numerical comparison operations i.e. $<$, $>$, $\leq$, $\geq$, $=$, can be represented.

Unfortunately numerical comparisons are more troublesome to implement than regular attributes. Conceptually it seems trivial to use numerical comparisons in access policies. Their result is boolean; the number either satisfies the comparison or it does not. Numerical attributes can be assigned to individual entities however in these access policies one needs to explicitly state ranges of numerical attributes i.e. all the attributes that are required. For example: an entity can be assigned a pay grade of four in a system with twelve levels, yet an access policy can state

$Paygrade < 4$ which, when using the previously mentioned access policy definition, would imply that the policy would be expanded as follows:

$$\text{Paygrade} = 3 \vee \text{Paygrade} = 2 \vee \text{Paygrade} = 1$$

In Bethencourt, Sahai and Waters [BSW07] the authors present a PBE scheme construction that supports numerical comparisons and provides a novel means by which to implement them. For all numerical attributes the authors re-represent them in base-2 form rather than in base-10. In this form *bit-masking* can be used to indicate preferred values, and positions of each individual binary digit within a binary string. Thus, an n-bit integer would be represented as $n$ attributes. For example using 4-bit binary strings, the attribute $a = 5$ would instead be represented as: `a:1***`, `a:*0**`, `a:**0*`, and `a:***1`.

When used in conjunction with disjunction and conjunction operations, the numerical comparison operation can be represented as an access policy in its own right. That is the operation can be represented as an access policy comprised of various bit masking values that when combined produce a value that satisfies the comparison. For example, the comparison $a < 11$ using 4-bit integers can be represented as follows:

$$\text{a:0***} \vee (\text{a:*0**} \wedge (\text{a:**0*} \vee \text{a:***0}))$$

The solution presented in Bethencourt, Sahai and Waters [BSW07] is a *natural* extension to the use of general predicates and fits in naturally with the existing definition. Numerical comparisons will be included when using general predicates within this thesis as a result.

#### Attribute Negation

Attribute negation can be seen as an important operation to perform upon an attribute. It is particularly useful when trying to exclude a certain member of a group where their presence may result in a conflict of interest. For example organising a surprise party for Bob who is leaving the Digital-Security research group. In this situation an ideal access policy for organisational messages would be:

$$\text{Digital-Security} \ \wedge \ \neg ID_{bob}$$

Where $\neg$ indicates attribute negation. Here Bob would not be able to access these secret messages and spoil the surprise. However, the addition of a negation operation is not trivial. Ostrovsky, Sahai and Waters [OSW07] looked towards, among other aspects, at implementing attribute negation. As such attribute negation will not be a supported operation within this thesis. More details concerning how attribute negation can be achieved can be found in Ostrovsky, Sahai and Waters [OSW07].

### 7.6.4 Examples

This section ends with an example to illustrate further the use and declaration of an access policy. The example is as follows:

$$(\text{Role:ceo} \vee (T_2(\text{Paygrade} = 3, \text{Group:audit}, \text{Role:secretary}) \wedge \text{OfficeNumber} = 1234))$$

With reference to Definition 2: The policy itself is the function $\mathcal{F}$ acting upon the set of attributes $\mathcal{P} = \{\text{ceo}, \text{Paygrade} = 3, \text{audit}, \text{secretary}, \text{OfficeNo} = 1234\}$. For the access policy to be satisfied the entity must either be a ceo, or work in Office 1234 and is either: a) a secretary or part of the audit team with a Paygrade equal to three; or b) a member of both the audit team and is a secretary.

## 7.7 Note on Existing Schemes

Numerous constructions of PBE are known to exist. Table 7.1 presents a categorisation of a select number of these constructions according to the characteristics as discussed in Section 7.1 and the expressiveness of their access policies. This is to give the reader some indication of the different constructions available and their differences at a high-level. Although the scheme presented in Bethencourt, Sahai and Waters [BSW07] is the only one that appears to support numerical operations, the use of numerical attributes and comparisons on said attributes can easily be extended to the other schemes during their implementation.

The papers listed also contain several variants of the main construction, found within each paper, that differ in relation to their implementation and assumptions made. The more mathematical oriented reader is asked to consult Chapter 8 for more information regarding the implementation of PBE schemes.

| **Construction** | **Family** | **Privacy** | **Access Policy Expressiveness** | | |
|---|---|---|---|---|---|
| | | | **Placement** | **Operators** | **Multi-Arg.** |
| [GS+06] | ABE | PH | KP | $\vee, \wedge, T_n(), =$ | yes |
| [PT+06] | ABE | PH | KP | $\vee, \wedge, T_n(), =$ | yes |
| [BSW07] | ABE | PH | CP | $\vee, \wedge, T_n(), <, >, \leq, \geq, =$ | yes |
| [Wat10, Appendix C] | ABE | PH | CP | $\vee, \wedge, =$ | no |
| [KSW08, Appendix C] | PBE | PH, AH | KP | $\vee, \wedge, T_n(), =$ | yes |

Table 7.1: Categorisation of several known PBE constructions

Other existing constructions not included in the categorisation include: Sahai and Waters [SW05]; Cocks [Coc01]; Yao, Fazio et al. [YF+04]; Gentry [Gen06]; Liang, Cao et al. [LC+09]; Ibraimi, Petkovic et al. [IP+09]; Shen, Shi and Waters [SSW09]; Lewko, Okamoto et al. [LO+10] and Bobba, Khurana and Prabhakaran [BKP10]. From which, the following presents some interesting findings:

**Shen, Shi and Waters [SSW09]** Introduces a symmetric variant of PBE that would allow for predicate privacy in a remote setting. For example, a user can upload their files to a remote server and query the server for files that satisfy a certain access policy using a specially constructed token. The server performing the query would not learn anything regarding the construction of the access policy.

**Bobba, Khurana and Prabhakaran [BKP10]** Extends upon the construction found in Bethencourt, Sahai and Waters [BSW07] by increasing the flexibility of representing user attributes in the keys.

Chapter **8**

# Constructing PBE Schemes

*The construction of Predicate Based Encryption schemes using general predicates is discussed. A CP-ABE from Waters [Wat10] is used to illustrate a working PBE scheme. A discussion over the security of PBE schemes is also provided.*

## 8.1 Overview

Chapter 7 provided a high-level overview over how PBE schemes work. Several existing schemes were discussed in Section 7.7. The aim of this chapter is to discuss how these PBE schemes can be realised.

As with many a encryption scheme, PBE schemes encrypt messages using some secret value. PBE schemes differ in their operation by using Linear Secret Sharing Scheme (LSSS) to distribute this secret value among a set of attributes according to some access policy. Only those authorised to decrypt the cipher-text will be able to reconstruct the secret value. While it is typical to see PBE schemes being implemented using pairing based cryptography, the techniques used to transform the predicate/access policy into a LSSS will vary. Moreover where this policy transformation occurs is dependent upon whether the scheme is a Key-Policy or Ciphertext-Policy scheme. To illustrate and discuss *all* the differences between the various methods of construction for different predicates only increases the scope and complexity of this discussion. As such the discussion within this chapter will only focus on general predicate PBE schemes. It is hoped by focusing on these schemes that the reader can gain a 'feel' for how other PBE schemes operate.

Constructions of PBE schemes can be characterised in terms of: a) how predicates are transformed into LSSSs; b) how the resulting LSSS integrates with the pairing based cryptography. c) the underlying pairing based cryptography. Section 8.2 offers a short introduction to pairing based cryptography and how the encryption and decryption operations can be realised. Section 8.3 provides an alternate and more formal definition for access policies, and how they relate to LSSS and cryptographic operations of PBE schemes. For many general predicate PBE schemes the means by which the LSSS and pairing based cryptography are integrated will differ substantially. Section 8.5 details how this can be achieved using the *large universe* CP-ABE scheme from Waters [Wat10]. Finally, Section 8.6 discusses several security aspects related to the construction of PBE schemes.

## 8.2 Pairing Based Cryptography

The construction given in Section 7.7 use pairing-based cryptography, specifically bilinear pairings, as a means to realise the encryption and decryption operations.

**Definition 3** (Bilinear Pairings)**.** *From Bethencourt, Sahai and Waters [BSW07]: Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}_0$ and e be a bilinear map, $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$. The bilinear map e has the following properties:*

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

2. Non-degeneracy*: $e(g, g) \neq 1$.*

*We say that $\mathbb{G}_0$ is a bilinear group if the group operation in $\mathbb{G}_0$ and the bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ are both efficiently computable. Notice that the map $e$ is symmetric since $e(g^a, g^b) = e(u, v)^{ab} = e(g^b, g^a)$.*

Bilinear pairings are used to perform the 'cryptographic step' used to encrypt a message [BF01]. The encryption and decryption operations can be defined, generally, as follows: When encrypting a message $M$ the encrypting entity will perform some 'encrypting' operation, say $\times$, on $m$ using a well known value $v$, i.e.

$$CT = M \times v.$$

This well known value is part of the recipients public key. During the 'decrypting' step the recipient will apply their 'decrypting' value, say $v^{-1}$, that reverses the original operation. The decrypting value is essentially the inverse of the encrypting value i.e.

$$M = CT \times v^{-1} = M \times v \times v^{-1}.$$

For example, take the well known *RSA* and *El Gamal* encryption schemes [RSA78; ElG85]. For *El Gamal* the transformation of a message $m$ to a cipher-text and back is:

$$CT = M \cdot s. \qquad M = CT \cdot s^{-1} = M \cdot s \cdot s^{-1}.$$

where $s$ is the shared secret shared between sender and recipient. For *RSA* the transformation is as follows:

$$CT = M^e \pmod{n}. \qquad M = CT^d \equiv (M^e)^d \pmod{n}.$$

where $e$ is the encryption exponent and $d$ is the decryption exponent that 'cancel' each other out because of the modulus operation. Within pairing-based cryptography the operation performed upon a plain-text message $M$, represented as a group element, is:

$$CT = Me(g, g)^s. \qquad M = \frac{CT}{e(g, g)^s} = \frac{Me(g, g)^s}{e(g, g)^s}.$$

where $e(g, g)$ is a bilinear mapping and $s$ is the secret exponent. More interested readers are asked to consult Dutta, Barua and Sarkar [DBS04]; Menezes [Men09] and Boneh [Bon07] for more information on pairing-based cryptography. The fine-grained access control associated with PBE comes from the use of LSSS schemes to distribute the secret exponent $s$ among group elements representing attributes. The security of pairing based cryptography is discussed in Section 8.6.4.

## 8.3 General Predicates and LSSS

As it was established within Section 7.1, predicates are used to specify for which sets of attributes decryption is permissible. This use of a general predicate is analogous to that of an *Access Structure*, a known mathematical construct. Formally, access structures are defined as follows:

**Definition 4** (Access Structure)**.** *Adapted from Beimel [Bei96, Section 3.2 Definition 3.5]: Let $\mathcal{P} = \{P_1, P_2, P_3, \ldots, P_n\}$ be a set of parties, an access structure $\mathcal{A}$ is the set of all authorised subsets of $\mathcal{P}$. Access structures are often* ***monotone****, that is given sets $Y$ and $Z$, if $Y \in \mathcal{A}$ and $Y \subseteq Z$ then $Z \in \mathcal{A}$ for all $Y, Z \subseteq \mathcal{P}$*

On their own access structures cannot be used directly within the cryptographic operations. Access structures must first be transformed into a Secret Sharing Scheme (SSS) scheme.

**Definition 5** (Secret Sharing Scheme (SSS))**.** *A SSS $\Pi$ is a scheme used to distribute a secret value $s \subseteq \mathcal{K}$, where $\mathcal{K}$ is a finite field, among a predefined set of participants $\mathcal{P}$ according to some access structure $\mathcal{A}$. The result of this distribution is that given an set of participants $B \subseteq \mathcal{P}$, if:*

- *$B \in \mathcal{A}$ then $s$ can be computed.*

- *$B \notin \mathcal{A}$ then $s$ cannot be computed.*

*For each scheme $\Pi$ there will exists a share (piece) generation and secret reconstruction function.*

While, SSSs can be used to distribute the secret, they imply that a fixed number of participants are required before the secret can be reconstructed. Threshold secret sharing differs from general secret sharing in that a minimum number of participants i.e. a *quorum*, is required before the secret can be reconstructed.

**Definition 6** (Threshold Secret Sharing)**.** *Given a set $S$ of possible secret values, a $(k, n)$-threshold scheme on $S$ is a (randomised) method of dividing each $s \in S$ into an array of shares $s_i \subset s$ such that:*

1. *Given any set of $k$ or more of the $s_i$, the secret value $s$ can be reconstructed easily.*
2. *Given any set of fewer than $k$ of the $s_i$, the secret value $s$ is completely undetermined in an information theoretic sense.*

Furthermore, a Threshold-SSS can also be linear, resulting in a LSSS.

**Definition 7** (Linear Secret Sharing Scheme (LSSS))**.** *Adapted from Beimel [Bei96]: A secret sharing scheme $\Pi$ is linear (a linear generation of pieces) if:*

1. *The piece of each party is a vector over $\mathcal{K}$.*
2. *During generation of the pieces, the dealer chooses independent random variables, denoted $r_1, \ldots, r_l$ each one distributed uniformly over $\mathcal{K}$. Each co-ordinate of the piece of every party is a linear combination of $r_1, \ldots, r_l$ and the secret $s$.*

Generally speaking: With pairing based cryptography each attribute is represented as a group element. By virtue of the *bilinearity* property it allows for two independent sets of operations to be performed upon a set of group elements representing each $P_i \in \mathcal{P}$. These operations hide the secret exponent among the group elements such that when the result of these operations are combined if the conditions are right the secret exponent to be recovered. These conditions are dictated by the LSSS.

The precise use of LSSSs to hide and recover the secret exponent is dependent not only upon the placement of the predicates within the PBE scheme but also upon the exact predicate used. For the remainder of this section, a general overview of how LSSSs are used as part of both CP and KP schemes. Section 8.5 provides a concrete example of how one can use LSSS precisely as part of a CP-ABE construction.

**Note.** *Several techniques that allow access structures to be turned directly into LSSSs are detailed within Section 8.4.*

### 8.3.1 Ciphertext-Policy

Recall from Section 7.3.2 that with CP schemes a message will be encrypted under an access policy $\mathcal{A}$ and can be decrypted from a key that is derived from a set of attributes $\mathcal{S}$. The use of LSSS within CP schemes can be seen as LSSS in its standard form.

**Encrypt** The LSSS is used to generate piece vectors, from the secret exponent, for each group element that represents an attribute $P_i \in \mathcal{P}$ as defined in $\mathcal{A}$. These vectors along with a description of the LSSS are stored along side the encrypted message.

**Key Generation** With the generation of decryption keys each user is assigned a set of attributes represented as a set of group elements modified by some random secret value.

**Decrypt** During decryption the LSSS, described in the cipher-text, will only reconstruct the secret exponent if an authorised set of attributes can be found within the elements of the decryption key.

### 8.3.2 Key-Policy

With Key-Policy schemes, a message will be encrypted under a set of attributes $\mathcal{S}$ and can be decrypted using a key that is derived from an access policy $\mathcal{A}$. The use of LSSS differs from its standard use.

**Encrypt** After the message has been encrypted the secret exponent is hidden among a set of group elements that have been derived from each $S_i \in \mathcal{S}$, the encryption key. These elements along with a description of $\mathcal{S}$ are stored along side the encrypted message.

**Key Generation** When generating the decryption key, the LSSS is used to distribute a secondary secret value among each attribute $P_i \in \mathcal{P}$ from $\mathcal{A}$. The resulting piece vectors and description of $\mathcal{A}$ are returned as the decryption key.

**Decrypt** With decryption the LSSS, taken from the decryption key, will only reconstruct the secret exponent if an authorised set of elements exists within the elements stored alongside the cipher-text.

## 8.4 Transforming an Access Policy into a LSSS

There are two families of technique that can be used to transform general predicates into a LSSS.

- *Access Tree*: Constructs a LSSS, based upon Shamir's construction [Sha79], directly from a tree representation of the access structure.
- *Monotone Span Program (MSP)*: Constructs a LSSS derived from a MSP, often referred to as the LSSS-Matrix technique.

The choice of technique when constructing a PBE scheme is important and will affect not only the supported operations that can be used when defining access policies (cf. Section 7.7) but also the efficiency of the implementation. It is known that implementations of LSSSs based upon *Shamir's* construction [Sha79], are not the most elegant way in which such schemes can be built. Beimel [Bei96] demonstrated the equivalence between a LSSS and a MSP. The LSSS-Matrix family of techniques presents a more efficient, and also interesting, construct. However, a downside with this technique is that the number of permissible arguments that each operator can have is dependent upon the precise algorithm used.

This section discusses three algorithms that can be used to transform an access structure into a LSSS by way of a MSP. These three algorithms are: a) *Lewko-Waters*; b) *Beimel*; and c) *Liu-Cao*. Before these algorithms are discussed some background information on MSPs is given, before the definition for a LSSS-matrix.

### 8.4.1 Preliminaries

*Span Programs* were first introduced in Karchmer and Widgerson [KW93], the following description of a span program is taken directly from Karchmer and Widgerson [KW93]:

> A *Span Program* is a linear algebraic model that computes a function $f$ on $n$ variable, over any field $K$. Given a fixed vector space $W$ over $K$, a non-zero vector $w \in W$ and let $X_i^\varepsilon$ (with $1 \leq n, \varepsilon \in \{0,1\}$) be $2n$ subspaces of $W$ that correspond to the $2n$ literals of $f$. Any truth assignment $\sigma$ to the variables makes exactly $n$ literals 'true'. We demand that the $n$ associated subspaces span the fixed vector $w$ iff $f(\sigma) = 1$.

More formally the definition for a Span Program is:

**Definition 8** (Span Program)**.** *From Karchmer and Widgerson [KW93, Definition 2]: Fix a field $K$. A span program over $K$ is a labelled matrix $\hat{M}(\mathcal{M}, \rho)$ where $\mathcal{M}$ is a matrix over $K$ and $\rho : rows(\mathcal{M}) \to \{x_i^\varepsilon \mid i \in [n], \varepsilon = 0,1\}$. The size of $\hat{M}$ is the number of rows in $\mathcal{M}$.*

When using span programs, for every input sequence $\sigma \in \{0,1\}^n$, the sub-matrix $\mathcal{M}_\sigma$ of $\mathcal{M}$ is defined by keeping the rows $r$ that are $\rho(r) = x_i^\varepsilon$ and $\sigma_i = \varepsilon$. The *span* of $\mathcal{M}$ is defined by the subspace generated by the rows of $\mathcal{M}$. It is denoted using $\mathsf{span}(\mathcal{M})$. Given the all one vector $\vec{1}$ in $W$, $\hat{M}$ accepts $\sigma$ iff $\vec{1} \in \mathsf{span}(\mathcal{M}_\sigma)$. Span programs can compute a boolean function $F$, if the program accepts exactly the inputs $\sigma$ where $F(\sigma) = 1$. Now the definition for a MSP can be given:

**Definition 9** (Monotone Span Program)**.** *Adapted from Karchmer and Widgerson [KW93, Section 3]: A span program $\hat{M}(\mathcal{M}, \rho)$ is called monotone, if the image of $\rho$ is only the positive literals $\{x_1, \ldots, x_n\}$. Such span programs can also compute monotone functions.*

By adapting the definitions for both LSSSs and MSPs a definition for an LSSS-Matrix can be constructed:

**Definition 10** (Linear Secret Sharing Scheme (LSSS)-Matrix)**.** *Adapted from Waters [Wat10] A secret-sharing scheme $\Pi$ over a set of parties $P$ is called linear (over $\mathbb{Z}_p$) if:*

1. *The shares for each party form a vector over $\mathbb{Z}_p$.*
2. *There exists a matrix $\mathcal{M}$ called the share-generating matrix for $\Pi$. The matrix $\mathcal{M}$ has $l$ rows and $n$ columns. For all $i = 1, \ldots, l$, the $i^{th}$ row of $\mathcal{M}$ we let the function $\rho$ define the party labelling row $i$ as $\rho(i)$. We consider the column vector $v = (s, r_2, \ldots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared, and $r_2, \ldots, r_n \in \mathbb{Z}_p$ are randomly chosen, then $\mathcal{M}v$ is the vector of $l$ shares of the secret $s$ according to $\Pi$. The share $(\mathcal{M}v)i$ belongs to party $\rho(i)$.*

**Remark.** *This is the definition for an LSSS-Matrix that has been used in several PBE constructions [GS+06; BSW07; Wat10]. When using such access structures, it is essential to remember that it is a MSP.*

**Note.** *For notation purposes the following shall be used to represent a LSSS access structure when describing PBE constructions: $(\mathcal{M}, \rho)$.*

### 8.4.2 Lewko-Waters Algorithm

First hinted in Beimel [Bei96, Chapter 4 Section 4.2 Example 4.5] and ultimately described in Lewko and Waters [LW10, Appendix G] this algorithm, based upon symmetric branching programs, requires that the access policy consists of two-argument conjunction and disjunction nodes and that leaf nodes represent the attributes of the policy. General threshold gates and multi-argument operators are not supported. An example of a supported formula is: $A \wedge (D \vee B \wedge\ C))$.

**Remark.** *Although, the algorithm was first hinted at in Beimel [Bei96] for the purposes of this document it shall be called the* Lewko-Waters *Algorithm.*

The *Lewko-Waters* Algorithm directly constructs the LSSS-Matrix from the access policy. This algorithm iterates over the nodes within the tree and labels each node with a vector whose construction is dependent on the type of node and its parent vector. Once each node has been labelled the vectors of the leaf node are used to construct the LSSS-Matrix.

**Note.** *In Liu and Cao [LC10, Section 1 Related Work] it is hinted that threshold gates contained within the access structure are first expanded to DNF form before use within the algorithm. Threshold gates are not supported directly. However, this has not been made explicit in the papers concerning this construction.*

### 8.4.3 Beimel Algorithm

This method as hinted in Beimel [Bei96, Chapter 4 Section 4.4] requires that the monotone boolean formula is first transformed into a LSSS and then into a monotone span program. To do so one can use the *Benaloh-Leichter* technique [BL88] to transform the boolean formula into a LSSS and then use the transformation method described in Beimel [Bei96, Chapter 4 Section 4.4 Claim 4.9] to transform the LSSS piece vectors into an MSP. Like the *Lewko-Waters* construction the boolean formula, taken as input, are represented as an access tree (or boolean circuit), however, unlike the previous technique, the interior nodes can be multi-input AND, OR and THRESHOLD gates. An example of a supported formula is: $(A \wedge C \wedge D) \vee (B \wedge C) \vee T_2(D, E, F)$.

Using this algorithm the size of the LSSS-Matrix is not optimal, the number of rows in the matrix is equal to the number of leaf nodes. Also Liu and Cao [LC10] mention that with threshold operators the number of rows within the LSSS-Matrix will be unnecessarily large. It is clear that LSSS-Matrices constructed using this technique are not optimal.

### 8.4.4 Liu-Cao Algorithm

In Liu and Cao [LC10], a recently released ePrint, the authors discuss and present an algorithm to construct optimal nay efficient LSSS-Matrices for use in CP-ABE, using an MSP based construction. They claim to be able to convert access trees in the same form as used with the aforementioned Lewko-Waters algorithm. For example given the following access structure:

$$\begin{aligned}\Gamma &= (A \wedge B) \vee (B \wedge C) \vee (A \wedge C)\\ &= (B \wedge (A \vee C)) \vee (A \wedge C)\\ &= T_2(A, B, C)\end{aligned}$$

And using the middle transformation as input, the Lewko-Waters algorithm generates the following LSSS-Matrix:

$$\text{matrix} := \begin{pmatrix} 1 & 1 & 0\\ 0 & -1 & 0\\ 0 & -1 & 0\\ 1 & 0 & 1\\ 0 & 0 & -1 \end{pmatrix} \ \rho := \begin{pmatrix} B\\ A\\ C\\ A\\ C \end{pmatrix}$$

Using the algorithm from Liu and Cao [LC10] they generate the following:

$$\text{matrix} := \begin{pmatrix} 1 & 1\\ 1 & 2\\ 1 & 3 \end{pmatrix} \ \rho := \begin{pmatrix} A\\ B\\ C \end{pmatrix}$$

It is clear that this matrix is smaller in size: Liu and Cao contend that when the boolean formula does not contain threshold gates, its size will be the same as that produced using the Lewko-Waters algorithm. If threshold gates are used then the Liu-Cao algorithm produces a smaller matrix. Given the recent publication status of the paper and that it was published as an ePrint leaves one with reservations concerning the papers claims i.e. it has yet to be peer-reviewed. However, once the authors claims have been substantiated then the *Liu-Cao* algorithm certainly does look promising.

## 8.5 Waters Construction

Section 8.2 detailed how pairing based cryptography performs the cryptographic step. Section 8.3 discussed how pairing based cryptography can be combined with a LSSS to ensure that only entities in possession of an authorised set of attributes can recover the secret exponent used to encrypt the message. Converting a predicate into a LSSS was discussed in Section 8.4. This section illustrates how all this can be combined to provide a coherent PBE scheme.

The "large-universe" construction (see Waters [Wat10, Appendix C]) has been chosen, it does not require any restriction on the size of $\mathcal{U}$ due to the use of a random oracle. This construction uses the *Decisional Parallel Bilinear Diffie-Hellman Exponent* (d-Parallel BDHE) as the main security assumption with respect to the pairing-based cryptography. The authors provide a Game based security proof for their construction, for brevity this scheme is given *sans* security proof. The full proof can be found in Waters [Wat10, Section 2.4.1, Section 3.1 and Appendix C.1]. The security of this scheme and other PBE is discussed more in Section 8.6.5.

### 8.5.1 Definition

**Definition 11** (Waters Ciphertext-Policy (CP) Attribute Based Encryption (ABE) Scheme). *Adapted from Katz, Sahai and Waters [KSW08, Section 3.1]: The Waters CP ABE scheme consists of four fundamental algorithms:*

- ***Setup**: takes as input a security parameter and outputs a master public key* MPK *(public parameters) and a master secret key* MSK*:*

$$(\mathsf{MSK}, \mathsf{MPK}) := \mathsf{Setup}(1^n) \tag{8.1}$$

- ***Encrypt**: takes as input a master public key, an access structure $\mathcal{A}$ and a message $M$ in some associated message space. It returns a cipher-text $CT$:*

$$CT := \mathsf{Encrypt}(\mathsf{MPK}, \mathcal{A}, M) \tag{8.2}$$

- ***KeyGen**: takes as input the master secret key and a set of attributes $S$ that describe the key. It returns the decryption key $\mathsf{Dec}(S)$:*

$$\mathsf{Dec}(S) := \mathsf{KeyGen}(\mathsf{MSK}, S) \tag{8.3}$$

- ***Decrypt**: takes as input the public parameters $\mathsf{MPK}$, a decryption key $\mathsf{Dec}(S)$ and a cipher-text $C$. It outputs either a message $M$ or the distinguished symbol $\perp$ if the set of attributes $S$ do not satisfy the access structure $\mathcal{A}$.*

$$\mathsf{Decrypt}(\mathsf{MPK}, \mathsf{Dec}(S), C) = \begin{cases} M & \textit{If correct private key} \\ \perp & \textit{If incorrect private key} \end{cases} \tag{8.4}$$

**Remark.** *The decrypt operation can also be defined without the public parameters and that their existence for decryption is implicit. In Bethencourt, Sahai and Waters [BSW07] the describe their system with the public parameters and when describing the implementation the public parameters are absent from the functions parameters.*

The remainder of this section will detail the implementations for each of the four algorithms specified in the definition.

### 8.5.2 Setup

The Setup algorithm (listed in Algorithm 1) takes as input a security parameter and outputs a master public key MPK (public parameters) and a master secret key MSK.

**Algorithm 1** Waters Setup Algorithm

```
Input: $1^n$ : Implicit Security Parameter
Output: MSK : Master Secret Key
Output: MPK : Master Public Key
1. Generate a prime $p$
2. Select $\mathbb{G}$ of prime order $p$ with generator $g$
3. Define a Hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{G}$
4. Choose random $\alpha, a \in \mathbb{Z}_p$
5. let MPK := $(g, e(g,g)^\alpha, g^a, \mathcal{H})$
6. let MSK := $g^\alpha$
7. return (MPK, MSK)
```

### 8.5.3 KeyGen

The KeyGen (listed in Algorithm 2) takes as input the master secret key and a set of attributes $S$ that describe the key. It returns the decryption key $\mathsf{Dec}(S)$.

**Algorithm 2** Waters Key Generation Algorithm

```
Input: MSK : Master Secret Key
Input: $S$ : Set of attributes
Output: Dec($S$) : Decryption Key associated with $S$
1. Choose random $t \in \mathbb{Z}_p$
2. let $K := g^\alpha g^{at}$
3. let $L := g^t$
4. let $\mathcal{K} := (\forall x \in S$ let $K_x := \mathcal{H}(x)^t)$
5. return $(K, L, \mathcal{K})$
```

### 8.5.4 Encrypt

The encryption algorithm works by first using the access structure to distributed the secret $s$ among the attributes and constructs a series of randomised variables to hold the result of this distribution. It then constructs the cipher-text and publishes that along with a description of the access structure. The algorithm is listed in Algorithm 3.

**Algorithm 3** Waters Encryption Algorithm

```
Input: MPK : Master Public Key
Input: $(\mathcal{M}, \rho)$ : An LSSS Access Structure
Input: $M$ : Plaintext
Output: $CT$ : Ciphertext
1. Choose random $s \in \mathbb{Z}_p$
2. // Distribute the secret exponent
3. let $\mathcal{M}$ be a $l \times n$ matrix
4. Construct random vector $\vec{v} = (s, y_2, \ldots, y_n) \in \mathbb{Z}_p^n$
5. for $i := 1$ to $l$ // For each row in matrix do
6.         let $\lambda_i := \vec{v} \cdot \mathcal{M}_i$
7.         Choose random $r_1, \ldots, r_l \in \mathbb{Z}_p$
8. end for
9. // Construct the Cipher-text
10. let $C := Me(g,g)^{\alpha s}$
11. let $C' = g^s$
12. for $i := 1$ to $n$ do
13.         let $C_i := g^{\lambda_i} \mathcal{H}(\rho(i))^{-r_i}$
14.         let $D_i := g^{r_i}$
15. end for
16. return $(C, C', (C_i, D_i))$ where $1 \leq i \leq n$ and a description of $(\mathcal{M}, \rho)$
```

**Remark.** *Recall: The function $\rho$ associates a row in $\mathcal{M}$ to an attribute. In this construction $\rho$ is an injective function and attributes are associated with at most one row in $\mathcal{M}$.*

### 8.5.5 Decrypt

Decryption of a cipher-text is relatively straight forward. First the algorithm checks to ascertain if the private key satisfies the access structure. If satisfaction does not occur then the algorithm returns the distinguished symbol $\perp$. Otherwise, the algorithm will identify the attributes that satisfied the access structure as: $\mathcal{I} \subset \{1, 2, \ldots, l\}$ where $\mathcal{I} = \{i : \rho(i) \in S\}$. The existence of $\mathcal{I}$ implies that there exists constants $\{\omega \in \mathbb{Z}_p\}_{i \in \mathcal{I}}$ such that if $\{\lambda_i\}$ are valid shares of the secret exponent $s$ then $\sum_{i \in \mathcal{I}} \omega_i \lambda_i = s$. These constants can be obtained using the LSSS associated with the access structure. The algorithm then uses these constants to reconstruct the secret exponent $s$ and will then perform a calculation using the value $C$ from the cipher-text that when divided out will return $M$. The algorithm can be found in Algorithm 4.

## 8.6 Security Discussion

This section discuss several security issues relating to the construction of PBE schemes in general and also relating to the Waters scheme illustrated in Section 8.5. For readers who are unfamiliar with formal cryptographic security the literature can be quite daunting, confusing and downright incomprehensible at times. This is due to a plethora of differing concepts and definitions arising from the various literature. Interested readers should consult Stinson [Sti04] for an interesting polemic on cryptographic security. As such this section of the thesis when dealing with more advanced concepts such as provable security and computational security will, at times, be sparse and shall include general descriptions only and where appropriate links to relevant further material.

**Algorithm 4** Waters Decryption Algorithm

```
Input: $\mathsf{Dec}(S)$ : Decryption Key
Input: $CT$ : Cipher-text
Input: $(\mathcal{M}, \rho)$ : Access structure associated with the Cipher-text, inclusion is implied.
Output: $M$ : Plain-text
1.  // Check for satisfaction
2.  if $\mathsf{Dec}(S)$ does not satisfy $(\mathcal{M}, \rho)$ then
3.          return $\perp$
4.  else
5.          let $\mathcal{I} = \{i : \rho(i) \in S\}$ where $\mathcal{I} \subset \{1, 2, \ldots, l\}$
6.          let $\{\omega \in \mathbb{Z}_p\}_{i \in I}$ be a set of constants obtained from $CT$.
7.          // Perform Decryption
8.          let
            $$\gamma := \frac{e(C', K)}{\prod_{i \in \mathcal{I}} (e(C_i, L)e(D_i, K_{\rho(i)}))^{\omega_i}} = \frac{e(g,g)^{\alpha s} e(g,g)^{ast}}{\prod_{i \in \mathcal{I}} e(g,g)^{ta\lambda_i \omega_i}} = e(g,g)^{\alpha s}$$
9.          return $\frac{C}{\gamma} = \frac{Me(g,g)^{\alpha s}}{e(g,g)^{\alpha s}} = M$
10. end if
```

### 8.6.1 Policy Expressiveness

The expressiveness allowed by an access policy is dependent on the transformation technique used to convert the predicate into a LSSS. The permissible node types and number of node inputs have been summarised within Table 8.1. From Table 8.1 it is clear that the Lewko-Waters technique differs in that it only permits operators with two arguments and does not support threshold operations natively. The remaining techniques do not possess such limitations.

| **Technique** | **Interior Node Type** | | | **Multi-input Gates** |
|---|---|---|---|---|
| | **OR** | **AND** | **THRESHOLD** | |
| Access Tree | yes | yes | yes | yes |
| LewkoWaters | yes | yes | not directly | no |
| Beimal | yes | yes | yes | yes |
| Liu-Cao | yes | yes | yes | yes |

Table 8.1: Summary of Access Policies Realisation and Permissible Boolean Formula Composition.

### 8.6.2 Entity Collusion

As the relationship between cipher-texts and entities, when decrypting, is one-to-many, one of the problems encountered when designing an PBE scheme is that of entity collusion. Entity collusion is an attack performed by two or more entities who individually do not have enough attributes to satisfy an access policy. The entities will then combine their attributes in an attempt to satisfy the policy. This is an important attack that constructions of PBE must be resistant to and many schemes have been designed to counter such an attack.

PBE schemes can be resistant to such attacks through the use of *blinding.* With blinding the secret exponent is blinding using a random value, say $\alpha$—the *blinding value* When decrypting a cipher-text the algorithm must recover the secret exponent as well as this blinding value i.e. $e(g,g)^{s\alpha}$. This value can be removed (blinded out) if the decryption key can satisfy the access policy. This random value and its distribution among the attributes is performed on a per entity basis. Malicious entities cannot combine their attributes and reconstruct the blinding value.

### 8.6.3 Complexity

The computational complexity of PBE schemes is dependent upon the exact construction of the scheme. Regardless, of exact construction some general observations over the complexity can be discerned. Below the complexity in terms of decryption key and cipher-text, and encryption and decryption times are discussed. For a more in-depth look at the complexity of several constructions of PBE the reader is asked to consult Emura, Miyaji et al. [EM+09].

**Size of Key(s) and Cipher-text** The size of the cryptographic keys and cipher-text within PBE schemes is inherently dependent on the number of attributes used during key construction. The size of the decryption keys and cipher-text will be based upon the bit-length of the group elements involved, if the construction is based upon pairings. The size of both MPK and MSK will remain constant during operation. For Ciphertext-Policy schemes the size of the cipher-text will be $\mathcal{O}(n+1)$, where $n$ is the *size* of (number of group elements used in) the access policy i.e. number of leaf nodes, and the one represents the element representing the encrypted message. Decryption keys will be $\mathcal{O}(\mid \mathcal{S} \mid)$. For Key-Policy schemes the size of the cipher-text will be $\mathcal{O}(\mid \mathcal{S} \mid +1)$ where the one represents the encrypted message. For both types of schemes the encryption keys can be represented a *plain-old-strings*.

**Computational Complexity** Related to the size of the cipher-text and is the computational complexity associated with the encryption and decryption steps. The following discussions apply to both Ciphertext-Policy and Key-Policy schemes.

**Encryption** The time required to perform encryption is dependent upon the number of exponentiations performed. For the Waters construction (see Section 8.5) the number of exponentiations required is $\mathcal{O}(n)$, where $n$ is the *size* of (number of group elements used in) the access policy i.e. its leaf nodes.

**Decryption** The time required is dependent upon: a) the number of pairing operations performed; and b) the minimum number of nodes required for the access policy to be satisfied. For the Waters construction the number of pairing operations required is $\mathcal{O}(P)$, where $P$ is the minimum number of nodes needed to satisfy the access policy.

### 8.6.4 Computational Security

The security of pairing-based cryptographic schemes relies upon the assumption made concerning the difficulty of computing some problem within some group model—see Dutta, Barua and Sarkar [DBS04, Section 2] for a partial list of these problems. For the Waters construction discussed, the authors specifically mention that they use a more formal assumption than the construction given in Bethencourt, Sahai and Waters [BSW07].

Within the construction given in Bethencourt, Sahai and Waters [BSW07] the authors rely upon the security of the generic bilinear group model (cf. Boneh, Boyrn and Goh [BBG05]) and also upon the use of random oracles. Bethencourt, Sahai and Waters argue that their construction itself is secure and that if a vulnerability were to be found then it must exploit specific mathematical properties of the generating elliptic curve or cryptographic hash function used. The assumption made in Waters [Wat10] is the authors own. The authors define and make use of the *Decisional Parallel Bilinear Diffie-Hellman Exponent* (d-Parallel BDHE) in conjunction with a random oracle to support a large universe.

When comparing both constructions the Bethencourt Sahai Waters (BSW) construction has been based upon the generic group model, though this assumption aides in an easier and more expressive construction it is nonetheless not 'ideal'. A more formal model should be given. The Waters construction provides an almost identical scheme concerning functionality and performance yet under a more stronger and formal proof model.

### 8.6.5 Provable Security

Aside from the computational security (see Section 8.6.4), *proof of security* for PBE schemes can be given. These proofs seek to define what it means for a cryptographic scheme to be secure, and the actions that a malicious entity (or entities) must take in order to compromise the scheme. The exact model used to define the security of the scheme is dependent upon the type of predicates used, access policy placement, if the scheme is attribute as well as payload hiding, and the security aims of the authors. Moreover, the proofs themselves are also dependant on the construction of the scheme itself. Two common approaches to constructing these models and proofs are that of *Game* and *Simulation* based approaches. However, these two approaches do have their differences [Den06, Section 2 c]. Below an overview of the game and simulation approaches are given together with their differences.

**Game-Based Approach** Game-based approaches looks at the interaction between an attacker and an instance of the cryptographic scheme, this interaction is the 'game' itself. This interaction is used to determine if the attacker is able to break the security of scheme. If it can be shown that the probability that the attacker is successful is small then security has been achieved. More interested readers are asked to consult individual constructions for differences in security models and Bellare and Rogaway [BR08, Introduction] for more information on Game based approaches. However, as noted in [Den06], a game based approach only looks at a single instance of the game and does not look at the security of the scheme in a larger context. Simulation is used to address this issue.

**Simulation-Based Approach** With simulation based approaches the goal is to look at the output from the crypto-scheme and compare it to the output from an idealised instance [GMW86]. Both an adversary and a representation of the environment in which the crypto-scheme is placed. The idealised version is used under the *proviso* that the idealised version of the crypto-scheme is unconditionally secure. If the output from the idealised version is *indistinguishable* from that of the non-idealised version, the non-idealised crypto-scheme is said to be secure. Hence, if the probability that a difference can be established between the outputs is small then the crypto-scheme is also secure. Although simulation based approaches presents a stronger security model the proofs are, inherently, more complex. For more information surrounding a simulation based security model, and example proofs, readers are asked to consult Katz, Sahai and Waters [KSW08] and Shen, Shi and Waters [SSW09].

Chapter 9

# Using PBE in Cryptographic Systems

*The use of Predicate Based Encryption as part of a cryptographic system is addressed. Issues arising from key management are treated. The deployment of both Key-Policy and Ciphertext-Policy schemes are illustrated. Finally, three modes of operation for Predicate Based Encryption schemes are introduced.*

## 9.1 Introduction

Cryptographic schemes document a set of related algorithms that when used facilitate some cryptographic primitive such as encryption or signing. On the other hand, cryptographic systems (crypto-systems) describe the deployment and subsequent use of various cryptographic primitives within some setting. When used as part of a crypto-system, public key encryption schemes provide guarantees towards message confidentiality **only** and allow for end-to-end security. Security requirements such as integrity, authenticity and non-repudiation require the additional use of digital signatures and other security mechanisms before any guarantees can be made.

On their own, traditional asymmetric crypto-scheme, such as *El Gamal* and *RSA* [RSA78; ElG85], cannot be used within a crypto-system. Their keys are simply *numbers* and as such without the use of a Public Key Infrastructure (PKI), the authenticity of encryption keys cannot be guaranteed. PKIs are the technical and organisational means required to generate and maintain asymmetric key pairs as used in a crypto-system Adams and Lloyd [AL02]. Normally PKIs are considered separately to encryption schemes and can be viewed as an add-on. However, decryption keys in PBE schemes can be derived either from: a) attributes that describe *real-life* characteristics; or b) access policies. The *right* of the decrypting entity to possess these keys needs to be proven. PBE schemes represent a mongrelised form of encryption scheme that requires elements from a traditional PKI such as key and certificate management before it can be successfully used at all, let alone as part of a crypto-system.

Unfortunately, current research towards PBE appears to have been focused on the construction of PBE schemes rather than the use of PBE as a part of a crypto-system. While this focus is not necessarily detrimental it has led to aspects such as key management and deployment to be assumed, weakly defined or not even addressed at all. This chapter aims to explore how PBE can be used as part of a crypto-system.

Section 9.2 provides a general overview over how PBE schemes can be deployed regardless of predicate placement. Management of a deployed PBE scheme is discussed in Section 9.3, before key management topics discussing key assignment (Section 9.4), key creation (Section 9.5 and key revocation (Section 9.6) are introduced. The effect predicate placement can have on the operation, and also deployment, of PBE schemes in Section 9.7. This chapter ends in Section 9.8 by introducing three modes of operation that can be applied to any PBE schemes.

## 9.2 General Model

Generally speaking, PBE is an asymmetric scheme and as such there are the traditional encrypting and decrypting entities that make use of the encryption and decryption algorithms from the underlying PBE scheme. However, PBE schemes require the use of a set of *master* keys to use and generate encryption and decryption keys, respectively. The creation and management of these master keys should be performed by an entity that can be considered, but not necessarily is, distinct from the encrypting and decrypting entities. Moreover, this entity will also be responsible for the creation of the master secret key. This additional entity will be referred to in this thesis as the Key Authority (KA).

Given the responsibility of decryption key generation it naturally follows that the KA is also responsible for the management of attribute universe $\mathcal{U}$. Encrypting entities will interact with the KA to acquire the necessary information to construct encryption keys and perform encryption. Decrypting entities, on the other hand, interact with the KA to acquire their decryption key. The role of the KA is discussed further in Section 9.3. Figure 9.1 provides a graphical representation of this general description, illustrating the distribution of the various algorithms from a PBE scheme among the different entities.

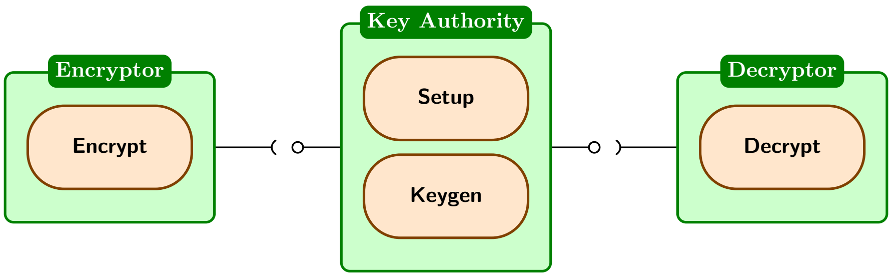


Figure 9.1: Distribution of cryptographic operations among entities.

From Section 7.2 it is known that PBE schemes have various characteristics that will affect how the scheme can be utilised. The expressiveness of the access policies will be determined by the underlying predicates used as part of the underlying PBE scheme. The characteristic of attribute hiding will affect the visibility of the attributes/predicates when they are embedded within the cipher-text. This will in turn affect the privacy and suitability of the PBE scheme used. The effect of predicate expressiveness and attribute hiding is nominal when discussing the overall operation of PBE within a crypto-system. Within this discussion no assumptions shall be made with respect to the expressiveness of access policies; all operations as described in Section 7.6 aside from attribute negation are assumed to be permissible. Similarly, whether the underlying scheme is attribute hiding or not will not be considered with regards to a systems operation.

The characteristic that has the greatest affect is that concerning predicate placement. Predicate placement gave rise to the Ciphertext-Policy and Key-Policy PBE scheme variants. These variants differ in relation to the access control afforded and will affect the semantics governing the attributes used to construct cryptographic keys. It follows that from these two distinct variants two distinct types of crypto-system can be constructed, one derived from a Ciphertext-Policy scheme and the other from a Key-Policy scheme. Although the effect of predicate placement is great there is nonetheless a fair degree of commonality between the different types of PBE scheme and their use as part of a crypto-system. In an effort to reduce the amount of needless repetition within this discussion the general model of PBE, as illustrated in Figure 9.1, will be referred to unless otherwise stated.

## 9.3 The Key Authority

The KA plays a central role within the operation of PBE. They are responsible for key management, that is the generation, issuing, and revocation of the cryptographic keys used by entities. Within PBE this will imply establishing the authenticity of the link between the semantics behind encryption and decryption keys, and their owners. This section explores the role that the KA has within PBE schemes.

### 9.3.1 Functions

The main functions of the KA are similar to those seen within a standard PKI. The main functions are listed below.

- **Master Key Set Generation**: The generation of both the master secret (MSK) and public (MPK) keys.

- **Cryptographic Key Issuance**: Decryption keys need to be generated and assigned to entities, with the entity's right to possess decryption key being verified. With encryption keys this entails controlling the ability of encrypting entities to both construct and use encryption keys.

- **Attribute Universe Management**: The attribute universe $\mathcal{U}$ needs to be defined, maintained and published.

- **Key Revocation**: When necessary attributes or access policies will need to be revoked. The act of revocation is based upon a decision to remove the item from circulation i.e. a decision has been made to purposefully curtail the lifetime of the item. The reasoning behind this revocation can be the result of either: a) an artificially induced lifetime e.g. weekly key cycle; b) the result of a managerial decision e.g. retirement; or c) a natural lifetime based upon an items semantic meaning.

- **Key Escrow and Recovery**: Key escrow, though, controversial, involve the KA maintaining a copy of an entity's decryption key that is made accessible to a higher authority. This may be as a result of organisational or legal practices. One of the benefits of key escrow is that it gives the KA a copy of an entity's decryption key in case the decryption key has been accidentally destroyed or its owner is no longer around. The downside is that a third-party is now in possession of an entity's decryption key, which can then be abused.

**Note.** *Although, key escrow can be classed as a main function, it can also be considered optional. Sending decryption keys upon creation to law enforcement agencies is not an explicit requirement in many a countries legal system.*

### 9.3.2 Components

Although, the KA can be depicted as a single entity, it is actual composed of specialist components that aide in achieving the functionality depicted in Section 9.3.1. These components can be considered as independent agents working together or as a single entity, and are described below and illustrated in Figure 9.2.

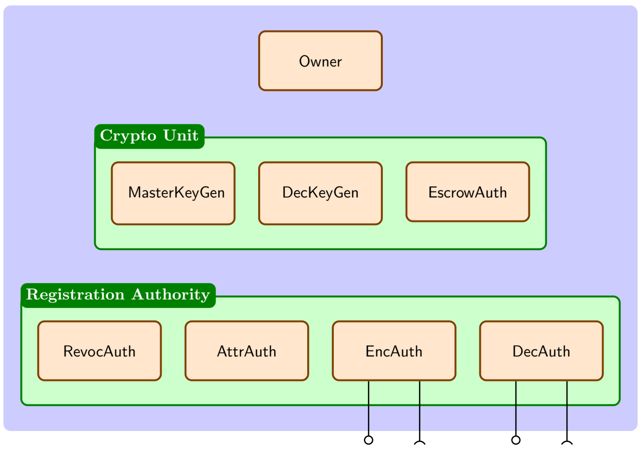


Figure 9.2: Components within a Key Authority

**Owner** The first component is that of the scheme owner and they are responsible for its operation.

**Cryptographic Unit** The next set of components are each involved with several cryptographic operations of the KA. These components may also each require that their hardware is physically secure, tamper proof and offers secure storage of data.

- Master Key Generator (MKG): The MKG is the centralised system responsible for generating and storing MPK and MSK. This component will use the setup algorithm from the PBE scheme.
- Decryption Key Generator (DKG): The DKG is used to generate decryption keys using the keygen algorithm.
- Escrow Authority (EA): The final component is that of the EA. This system is responsible for keeping copies of generated decryption keys for retrieval at a later date. Naturally, access to the decryption keys should be strictly controlled and logged.

**Registration Authority** The final set of components are responsible for managing the operation of the KA. These components can be seen as the main interface to the KA through which entities interact.

- Attribute Authority (AA): The AA is the component responsible for the management of the attribute universe $\mathcal{U}$. Management of $\mathcal{U}$ includes, per attribute its: definition, propagation, assignment, revocation, and canonicalisation.
- Encryption Authority (EncAuth): The EncAuth is responsible for issuing the master public key to authorised entities and information relevant to the construction of encryption keys. This is an encrypting entity's main point of contact with the KA.
- Decryption Authority (DecAuth): The DecAuth is responsible for the generation of decryption keys and their correct assignment to entities. This is a decrypting entity's main point of contact with the KA.
- Revocation Authority (RA): The RA is responsible for specifying and enforcing the revocation of attributes and access policies. This component may also require the use of a secure means of generating time-stamps.

## 9.4 Issuing Cryptographic Keys

### 9.4.1 Decryption Keys

Decryption keys are generated by the KA for use by the decrypting entity. This leaves the source of key composition open. The input for decryption key generation can either stem from the KA themselves, or from the decrypting entity. This would be an implementation choice, either decryption keys are assigned or they are requested. However, this would imply that the decrypting entities have a say in their access control which unless vetted is not preferable. Regardless, the link between decryption key semantics and the decrypting entity must first be verified. The means by which verification is performed is context dependent and will vary. The steps involved are outlined below.

**Step** 1: *Key Request*: First the decrypting entity sends a decryption key generation request to the DecAuth along with information concerning their identity.

**Step** 2: *Identity Verification*: The DecAuth will then verify the identity of the entity.

**Step** 3: *Information Collection*: The DecAuth then collects the information required to generate the key from the AA. It is assumed that based upon the verified identity that the AA will collect the correct information. This includes key revocation information.

**Step** 4: *Key Generation*: With the validated information, the DecAuth will send this information to the DKG. After generating the decryption key, the DKG returns the key to the DecAuth.

**Step** 5: *Key Escrow*: Depending on the operation policies of the PA, the decryption key may be stored with the EA after it is generated.

**Step** 6: *Key Issuance*: This final step concerns the DecAuth sending the decryption key together with MPK securely to the requesting entity.

### 9.4.2 Encryption Keys

The generation of encryption keys is more interesting when compared to the generation of decryption keys. While, the generation of decryption keys was performed mainly by the KA, encrypting entities should have a say in the composition of the encryption keys—it is only natural. The composition of the encryption key is technically arbitrary and the attributes chosen by the encrypting entity may not exist within the attribute universe as defined by the KA. This presents the issue of who constructs encryption keys. There are several solutions to consider, they are:

**Solution** I: The KA may produce authorised encryption keys that are requested by encrypting entities. This is similar to how the decryption keys are generated but is restrictive and does not allow the encrypting entity to have an encryption key that is precisely what they want.

**Solution** II: The KA may actively restrict the set of recognised attributes used to generate encryption keys. The KA publishes the information required to construct keys allowing the encrypting entities some freedom in constructing their keys.

**Solution** III: The KA will publish the information required to construct encryption keys, and also allow the encrypting entities to submit new attributes for inclusion in $\mathcal{U}$ alongside information on any restrictions associated with them.

With encrypting key generation the overriding theme is that of control, with Solution I, the KA is in complete control over the specification of the encryption key. With Solution II, the KA is in less control and the encrypting entity has some control. With Solution III, the encrypting entity has complete control. While the first two solutions look appealing there is still the chance that encryption keys will be used that use attributes not in $\mathcal{U}$, moreover, with Solution I, the exact composition of the encryption key is known by the KA, this raise issues concerning the maliciousness of the KA—discussed further within Section 10.4. This leaves Solution I, as the more viable one.

The information required to construct encryption keys is subject to the underlying PBE scheme type. Key-Policy requires knowledge of $\mathcal{U}$ and $\mathsf{MPK}$ and Ciphertext-Policy requires the same as well as knowledge of the predicate type—this allows for syntactically correct predicates to be constructed. Moreover, the KA can restrict the ability of an encrypting entity to use encryption keys by restricting access to $\mathsf{MPK}$. Furthermore, the onus is on the encrypting entity with regards to checking the encryption key for any defects and ensuring that any newly defined attributes, together with information concerning their revocation and who they can be issued. The steps for 'issuing' encryption keys are:

**Step** 1: *Key Request*: First the encrypting entity sends a request to the EncAuth along with information concerning their identity.

**Step** 2: *Identity Verification*: The EncAuth will then verify the identity of the entity.

**Step** 3: *Information Collection*: The DecAuth then collects the information required to generate encryption keys from the AA, MKG i.e. the $\mathsf{MPK}$, and the RA.

**Step** 4: *Key Issuance*: This final step concerns the DecAuth sending the information securely to the requesting entity.

## 9.5 Constructing Keys

Within PBE, cryptographic keys are either a set of attributes, or an access policy. When construction keys comprised of a set of attributes their construction can be seen as trivial, it is the selection of the attributes that is of more concern. Recall from Section 9.4 that it was discussed that this process is verified by the user of the attributes. However, with cryptographic keys made from access policies, the definitions given in Section 7.6 do not account for their management and use. So far in this thesis no formal model has been described that can be use to aid in the creation and definition of access policies. A formal model provides for a more esoteric understanding towards the operation of PBE out with the confines of predicate placement and attribute placement. Furthermore, a formal model allows for the semantic use of access policies to reasoned upon and also defined. This also has a bearing when discussion the construction and definition of the attribute universe, and to some degree the selection and definition of the set of attributes used as cryptographic keys

From Section 7.4 it is known that PBE facilitates ABAC. As such existing policy models for ABAC systems can be used as a guide (and base) when defining a formal access policy model for PBE schemes. Specifically, the model expressed in Yuan and Tong [YT05, Section 3.2] can be used directly , albeit with a few modifications, as a policy model for PBE schemes. This model is presented in the next section.

### 9.5.1 Formal Model

**Attribute Universe** From Section 7.5, a grouping of attributes were given in terms of their use to describe, entities, resources and environments. Formally these groupings are represented as follows:

$$\begin{aligned}\mathcal{S} =& S_1 \times \cdots \times S_I \\ \mathcal{R} =& R_1 \times \cdots \times R_J \\ \mathcal{E} =& E_1 \times \cdots \times E_K\end{aligned}$$

Where, $S_i, R_j, E_k$ represent an attribute that is used to describe an entity, resource and environment, respectively. Following this the attribute universe is defined as $\mathcal{U} := \mathcal{S} \times \mathcal{R} \times \mathcal{E}$.

**Attribute Relations** Given an entity $s$, resource $r$ and an environment $e$, $\text{ATTR}(s) \subseteq \mathcal{S}$, $\text{ATTR}(r) \subseteq \mathcal{R}$, $\text{ATTR}(e) \subseteq \mathcal{E}$ represent the attribute assignments for $s, r$ and $e$ respectively. For attributes that can possess different values, such as *Role*, function notation is used for value assignment. For example: $\text{Role}(s) = Manager$.

**Policy Rule** A policy rule is a function that when given an entity $s$ wishing to decrypt a cipher-text $r$ in an environment $e$ determines if the entity is allowed to decrypt or note. A rule is described as a boolean function $\mathcal{F}$ operating on the attributes associated with $s, r$ and $e$:

$$Policy(s, r, e) \leftarrow \mathcal{F}(\text{ATTR}(s), \text{ATTR}(r), \text{ATTR}(e))$$

If the boolean function $\mathcal{F}$ evaluates to true then decryption will be permitted, otherwise it is not. It is from these rules that access policies used as cryptographic keys can be constructed.

**Note.** *Policy rules can, though defined separately, can also be merged together to form new policy rules.*

**Policy Store** A *Policy Store*, represents a collection of policy rules that are used to mitigate access to resources by entities within a particular environment. It is from this store that the access policies, as generated from, policy rules are to be found.

### 9.5.2 Using

Some examples are given to illustrate the composition of policy rules and the extraction of access policies from them. Policy rules can come in various guises, dependent upon the context of their use. One common form seen is that of name, value pairs in which attributes assigned to resources, entities and the environment are not related. The example access policy given in Section 7.6.4 can be derived from one such policy rule. This rule is as follows:

$$\begin{aligned}Policy(s, r, e) \leftarrow& (\text{Role}(s) = ceo) \vee ((\text{OfficeNumber}(s) = 1234) \\ & \wedge T_2(\text{Paygrade}(s) = 3, \text{Group}(s) = audit, \text{Role}(s) = \text{secretary}))\end{aligned} \tag{9.1}$$

The description of the access control afforded by this rule is found within Section 7.6.4. Entities are assigned attributes that describe their *Role(s)*, *Paygrade*, *Groupings* and *Office Number*. Implying that the set of attributes used as a key would be comprised of the aforementioned descriptive attributes.

**Remark.** *With respect to attribute canonicity (see Section 7.5), the functional notation that indicates value assignment can be used as a guide when declaring the value of the attributes as used by the underlying cryptography.*

The model presented in Yuan and Tong [YT05], facilitates much more richer access control semantics than simple name value pairs. The attributes of resources and entities, and environment can be used together to specify interesting access control rules. For example:

$$\begin{aligned} Policy(s,r,e) \leftarrow & (\text{Group}(s) = \text{Group}(r)) \\ & \wedge(\text{HireDate}(s) \leq \text{CreationDate}(r)) \\ & \wedge(\text{CurrentDate}(e) \leq 20121212) \end{aligned} \tag{9.2}$$

This policy states that access is granted if a) the entity and resource have the same group attribute; b) the entity was hired before the creation of the resource; and c) the current date is before December, $12^{th}$ 2012. Recall that policy rules are used directly as access policies. Specific instances of this policy rule are used as access policies. For example:

$$\text{Group:it-team} \wedge (HireDate \leq 20101212) \wedge (CurrentDate \leq 201212)$$

Here the 'Group' attribute has been specified as that of the `it-team` and that the creation date of the resource is December, $12^{th}, 2010$.

When placed within the context of a crypto-system, authority over the policy store and the creation of policy rules lie the KA (within Key-Policy schemes) and the encrypting entities themselves—within Key-Policy schemes. This will have a bearing on the access control offered. This is discussed further, and *per* scheme type, in Section 9.7.

## 9.6 Key Revocation

Within the context of PBE key revocation is applicable to both access policies and attributes, the two types of cryptographic *key* found. The reasoning behind revocation can be the result of either: a) an artificially induced lifetime e.g. weekly key cycle; b) the result of a managerial decision e.g. retirement; or c) a natural lifetime based upon an items semantic meaning. Either way, cryptographic keys will possess some form of lifetime and the problem in relation to PBE is: *How can a lifetime be enforced upon attributes and access policies?* Unfortunately, revocation mechanisms for PBE schemes have not been studied in too great a depth. Fortunately, the revocation of attributes is a known problem within IBE systems [Sha85; BF01], and these techniques can be carried over into PBE schemes—Bethencourt, Sahai and Waters [BSW07, Section 4.3]. Although it must be stated that these techniques by no means represent an absolute solution.

### 9.6.1 Some Solutions

The solutions seen within IBE schemes stipulate that an expiration date be appended to each attribute. When applying this solution to PBE there are two ways in which it can be implemented, either as: a) a textual attribute; or as b) a numerical attribute. With the latter imploring the use of numerical comparisons. Through use of an expiration dates attributes now possess a fixed life time and are usable until their date of expiration. However, these same attributes cannot be pulled from circulation and are still usable after they have been expired. Proper attribute revocation i.e. removing an attribute from circulation, is not permissible.

**Textual Solution** With the textual solution, attributes that possess a limited lifetime will take the form: `<attribute>:<date>`, where `:` denotes concatenation and date is the expiration date to be used. When the expiration date passes a new attribute will be issued as a replacement. When constructing access policies, a date attribute can be used as a clause surrounding the access policy e.g. $date \wedge (\ldots)$. The conjunction stipulates that an entity needs to be in possession of the date attribute as well as satisfy the internal access policy. However this is itself not a perfect solution, dates are treated as textual attributes and specify the date that the attributes are valid until. If an access policy is to be created whose use spans multiple lifetimes of an attribute, clauses need to be added to support each lifetime. A range of dates can not be specified. Furthermore, the format of the date itself would need to be finalised and propagated by the KA.

With respect to the validity period of an attribute, the more fine-grained the validity period, the higher the frequency of attribute renewal. This will: a) increase the interactions between the KA and entities; and b) increase the number of attributes that need to be managed, and supported for both decrypting and encrypting entities.

**Numerical Solution** An alternative solution is to treat the date attribute as a number in big endian form, that is the date: February, $29^{th}$, 2009, would be represented as 20090229. It is clear that with this solution the granularity of the date would need to be established in accordance with the renewal period i.e. daily, monthly, annually *et cetera*. Attributes, when assigned an expiration date, will thus take the form of key-value pairs i.e. `<attribute>=<date>`. Unfortunately with attributes that are already in a key-value form, the use of a date in this fashion cannot be so, the attribute already has a 'value'. In such circumstances appending the date to the attribute name itself would be a one solution. With a key-value form for attributes, it allows these date values to be queried and also for ranges to be specified. For instance, given an attribute representing a student within the academic year starting 2010 their *student* attribute would be: Student = 2010. To specify that access is for all students enrolled between the years 2000 and 2010, the following policy can be created: Student >= 2000 $\wedge$ Student <= 2010. Similarly, one can specify that access is for all current students enrolled since 2000 the policy is: Student >= 2000. Treating the date as a number allows for: a) periods to be specified in a compact manner; and b) decreases the complexity when managing access policies. However, entities whose keys are comprised of attributes still bare the burden of managing specific instances of these attributes.

## 9.7 Types of Crypto-System

The discussion within this chapter has been concentrated on looking at PBE schemes in general when used within a crypto-system. This was due to the large degree of commonality between Ciphertext-Policy and Key-Policy oriented PBE schemes. This section looks at these two types of schemes individually in terms of their operation, access control and example usage.

**Remark.** *Section 9.2 already provided an outline of the interaction between entities and the KA. This was further refined in Section 9.4 detailing the steps taken by entities to obtain cryptographic keys. Here the operational outline will tie together the information from the previous sections together with explicit details over the information being passed around.*

### 9.7.1 Ciphertext-Policy Crypto-Systems

Within Ciphertext-Policy PBE schemes access policies are used to encrypt data and sets of attributes are used to derive decryption keys. Figure 9.4a outlines the interactions between the decrypting and encrypting entities, with the KA. The steps for encryption and decryption are as follows:

**Encrypting Data** The steps to encrypt a message $M$, are:

**Step** 1: The encrypting entity identifies and authenticates itself with the KA.

**Step** 2: In return, the KA sends a) $\mathsf{MPK}$; and b) $\mathcal{U}$ together with the operations permissible. $\mathsf{MPK}$ indicates that the encrypting entity has the right to encrypt data. The remaining information allows the encrypting entity to construct policy rules from which access policies can be generated.

**Step** 3: With the required information, the encrypting entity can then construct a policy rule and derive the necessary access policy $\mathcal{A}$.

**Step** 4: The final step is the use of $\mathsf{MPK}$ and $\mathcal{A}$ to generate a cipher-text $CT$.

**Decrypting Data** The steps to decrypt a cipher-text $CT$ are:

**Step** 1: The decrypting entity identifies and authenticates itself with the KA.

**Step** 2: The KA constructs a set of attributes $\mathcal{S}_{entity}$, derived from the decrypting entity's identity $ID_{entity}$.

**Step** 3: $\mathcal{S}_{entity}$ is then used to generate a decryption key $\mathsf{Dec}(\mathcal{S}_{entity})$.

**Step** 4: $\mathsf{Dec}(\mathcal{S}_{entity})$ and MPK are then sent to the decrypting entity.

**Step** 5: Using $\mathsf{Dec}(\mathcal{S}_{entity})$ and MPK the decrypting entity can then decrypt $CT$.

## Access Control

Ciphertext-Policy schemes provide a tight fit with the ABAC model as described within Section 7.4. The decrypting entity i.e. the subject, is assigned attributes that are used to satisfy an access policy, that is attached to the cipher-text i.e. the resource. This mimics closely the operation of ABAC in that the entity comes to the cipher-text with a set of attributes that will decided if access is granted. With access policies being attached during encryption, the access policy governing a cipher-text cannot be modified at a later date without re-encrypting the data. The decision to decrypt lies with the cipher-text rather than the decrypting entity. As such the access policy, used for encryption, is essentially *tied* to the cipher-text, and all decrypting entities are subject to this access policy. For decrypting entities, this is a permissive environment, and their decryption keys place no restrictions *per se* on what they can and cannot decrypt. It is the cipher-text, through the access policy used for encryption, that places the restrictions. Again this mimics the access control afforded by ABAC, moreover it is reminiscent of the access control as seen in RBAC models [SC+96]. Within this setting the emphasis, for access, lies with whom the decrypting entity is and what they are, rather than with the resource itself: access is user-centric.

With respect to the authorisation architecture, the AA resides with the Key Authority. The KA is responsible for the assignment, and specification, of attributes to entities. Thus the KA is also partly responsible for the actions associated with a PDP. The actions of the PDP and the PA lie with the encrypting entity. The encrypting entity both specifies, constructs and uses the access policy. That is the encrypting entity has their own policy store populated with various policy rules. Alternatively, the operations associated with the PA and PDP may not lie with the encrypting entity and instead lie with the KA. The encrypting entity would only indicate the intended recipients. Finally, the PEP is associated with the decrypting entity, the act of decryption is in itself a *fixed point* where the right of the decrypting entity is challenged and the result will determine is decryption occurs or not.

**Remark.** *The user-centric nature of the access control implies that the attributes described in the attribute universe $\mathcal{U}$ will be used primarily to describe the decrypting entities i.e. their identity, rather than the data itself.*

## Example

With the user-centric nature of Ciphertext-Policy schemes, an example setting for its use would be that of a University. Here the university would play the role of the KA, and students (and university staff) would be the encrypting and decrypting entities. Upon matriculation, students are assigned attributes that represent their matriculation number, status as a student, and degree programme into which they have enrolled. During the course of their studies, students will be assigned attributes representing the courses that they have enrolled in to, and also any official affiliation within the university itself.

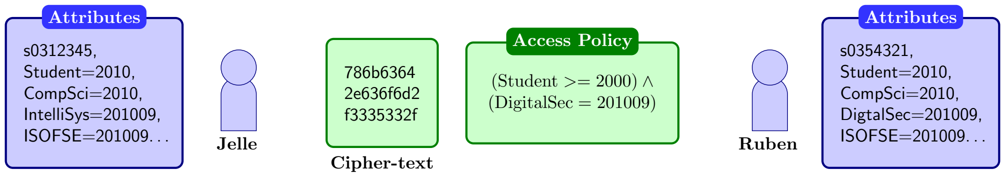
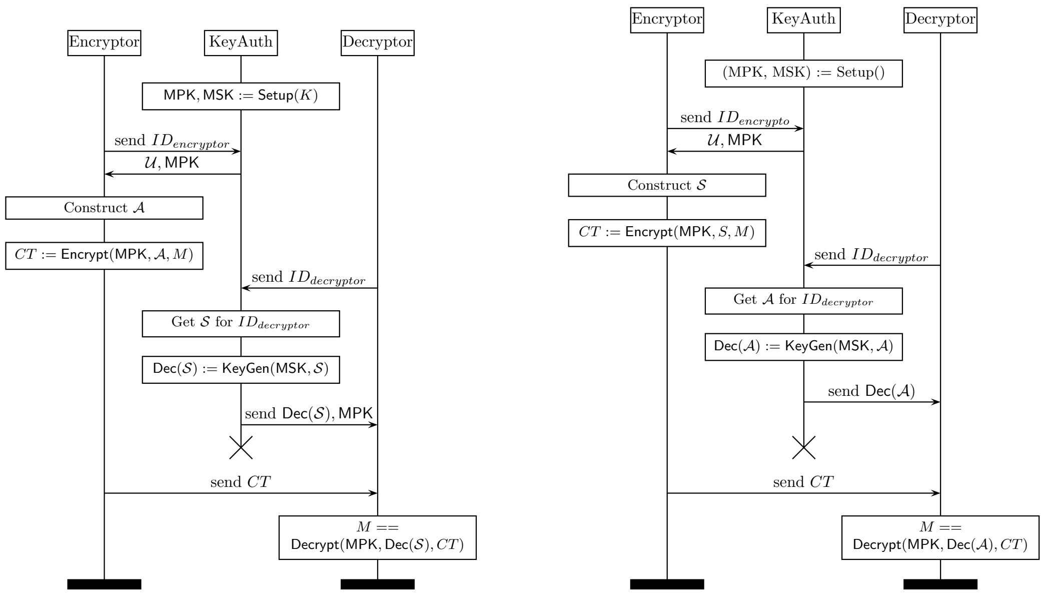


Figure 9.3: Example of cryptographic keys for Ciphertext-Policy systems.

With respect to key revocation, recall from Section 9.6.1 the representation of dates as numerical attributes. In this example, the attributes representing status and degree programme would be renewed each year at matriculation, indicating the attributes are to be renewed annually and their value is the current year in which the academic year began. For attributes representing official affiliations and course enrolment the granularity of renewal is per semester. While it is tempting to treat the semester values as ‘1’ and ‘2’, the year must also be taken into account.


Encryptor
KeyAuth
Decryptor
MPK, MSK := Setup($K$)
send $ID_{encryptor}$
$\mathcal{U}$, MPK
Construct $\mathcal{A}$
$CT$ := Encrypt(MPK, $\mathcal{A}$, $M$)
send $ID_{decryptor}$
Get $\mathcal{S}$ for $ID_{decryptor}$
Dec($\mathcal{S}$) := KeyGen(MSK, $\mathcal{S}$)
send Dec($\mathcal{S}$), MPK
send $CT$
$M$ == Decrypt(MPK, Dec($\mathcal{S}$), $CT$)
Encryptor
KeyAuth
Decryptor
(MPK, MSK) := Setup()
send $ID_{encrypto}$
$\mathcal{U}$, MPK
Construct $\mathcal{S}$
$CT$ := Encrypt(MPK, $S$, $M$)
send $ID_{decryptor}$
Get $\mathcal{A}$ for $ID_{decryptor}$
Dec($\mathcal{A}$) := KeyGen(MSK, $\mathcal{A}$)
send Dec($\mathcal{A}$)
send $CT$
$M$ == Decrypt(MPK, Dec($\mathcal{A}$), $CT$)

(a) Ciphertext-Policy (b) Key-Policy

Figure 9.4: Operational Outline for PBE Systems

Affiliations, and course taken, change from year to year. Here the value of the attribute would be comprised of the year and the numerical value of the month in which the semester began. The attribute used to represent matriculation number needs no revocation value as matriculation numbers are unique per student. An example set of these attributes can be found within Figure 9.3. An example policy rule from which access policies can be constructed is:

$$\begin{aligned} Policy(s, r, e) \leftarrow &(\text{Student}(s) = \text{CurrentYear}(e)) \\ &\wedge (\text{Group}(s) = \text{CurrentSemester}(e)) \end{aligned} \tag{9.3}$$

This policy rule states that only students within the current year, and who are also affiliated with a group within the current semester. Figure 9.3, provides an example of an access policy derived from this policy rule. The access policy states that only students from the last ten years **and** who are affiliated with the Digital Security group within the first semester of the academic year 2010 are permitted access—Ruben within the example.

### 9.7.2 Key-Policy Crypto-Systems

In Key-Policy PBE schemes access policies are used to construct decryption keys and sets of attributes are used as encryption keys. Figure 9.4b outlines the interactions between the decrypting and encrypting entities with the KA. The steps for encryption and decryption are:

**Encryption** The steps required to encrypt a message $M$ are:

**Step** 1: The encrypting entity identifies and authenticates itself with the KA.

**Step** 2: In return, the KA sends: a) MPK; and b) $\mathcal{U}$. As before, MPK indicates that the encrypting entity can encrypt data, and $\mathcal{U}$ allows the entity to construct sets of attributes.

**Step** 3: Using $\mathcal{U}$, the encrypting entity can then construct a set $\mathcal{S}_M$ of attributes to encrypt $M$ under.

**Step** 4: Finally $M$ is encrypted using MPK and $\mathcal{S}_M$.

**Decryption** The steps required to decrypt a cipher-text $CT$ are:

**Step** 1: The decrypting entity identifies and authenticates itself with the KA.

**Step** 2: The KA can then construct an appropriate access policy $\mathcal{A}_{entity}$, for the decrypting entity.

**Step** 3: $\mathcal{A}_{entity}$, is then used to generate a corresponding decryption key $\mathsf{Dec}(\mathcal{A}_{entity})$.

**Step** 4: $\mathsf{Dec}(\mathcal{A}_{entity})$ and $\mathsf{MPK}$ are then sent securely to the decrypting entity.

**Step** 5: $\mathsf{Dec}(\mathcal{A}_{entity})$ and $\mathsf{MPK}$, the decrypting entity can then decrypt $CT$.

### Access Control

With Key-Policy schemes, the mimicry of ABAC is not as tight when compared to Ciphertext-Policy schemes. The decrypting entity i.e. the subject, is assigned an access policy that is satisfied by the attributes attributed to the cipher-text i.e. the resource. It is not the decrypting entity that provides the attributes but the resource. Access policies are constructed per decrypting entity and explicitly states the cipher-texts the decrypting entity can and cannot decrypt. It is the decryption key that places restrictions on what is required for decryption to occur. This is a permissive environment for decrypting entities and somewhat restricts what they can do i.e. decrypt, by default. Such functionality is more akin to *Mandatory Access Control* models [SS94] than with ABAC. Implying that the access control afforded by Key-Policy schemes are preferable in proscriptive environments. Moreover, as the attributes are attributed to the cipher-text rather than the decrypting entity, it also gives the impression that Key-Policy schemes are more suitable in data-centric environments in which the data itself is more important than the identity of the decrypting entity.

Concerning the authorisation architecture, as per usual, the AA resides with the Key Authority. As with CP schemes the PEP is associated with the decrypting entity. The responsibility for actions associated with the PA and PDP also lies with the KA. The KA will have its own policy store containing various policy rules from which to construct access policies used to construct decryption keys. Although the encrypting entity can specify the attributes used to construct the encryption key, ultimately it is the KA with whom access is decided. This is exacerbated further with the alternative idea of the KA constructing $\mathcal{S}$ using input from the encrypting entity: the KA **is** in control.

**Remark.** *As access policies are* not *constructed by the encrypting entity it presents an interesting dilemma, the encrypting entity is reliant upon the KeyServer to adhere dutifully to its remit. Moreover, the data-centric nature of the access control implies that the attribute universe $\mathcal{U}$ will be used primarily to describe resources.*

### Example

The setting of mitigating access to a set of `TCP/IP` connections provides a setting that Key-Policy schemes would excel in. An *Internet Service Provider*, acting as the KA would mitigate access to traffic logs to authorised third-parties such as law enforcement agencies. When a connection is logged PBE is used to encrypt the payload. The set of attributes used for encryption can be derived from the `TCP` and `IP` headers. One possible set of attributes includes: a) the source and destination, port numbers and `IP` addresses; and b) timestamps corresponding to connection establishment and termination. Concerning key revocation, both the connection and termination times associated with the connection will be represented as numerical attributes i.e. key-value pairs. This representation shall also be used for the `IP` addresses, each resolved to single number, and port numbers. Figure 9.5 contains an example of a set of attributes associated with several connections. With this example, the policy rules can be used to specify the subset of the logs that the decrypting entity will be able to decrypt. An example of one such policy rule is:

$$\begin{aligned} Policy(s, r, e) \leftarrow & (\text{DestinationAddress(r)=722620399}) \\ & \land ((\text{DestinationPort}(r) <= 8080)) \\ & \land (\text{DestinationPort}(r) >= 8000)) \end{aligned} \tag{9.4}$$

This policy rule states that access is granted if the destination `IP` address is `72.26.203.99` and the destination port should be between 8000 and 8080. Figure 9.5 contains the access policy derived from this rule.

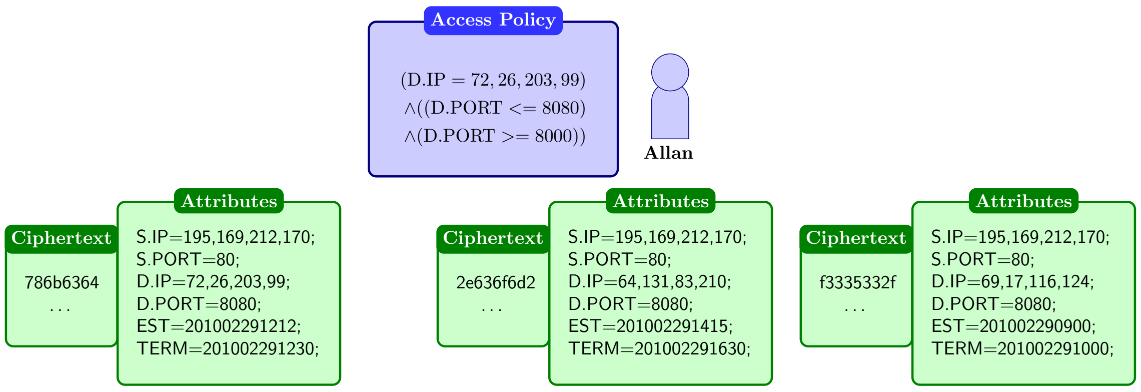


Figure 9.5: Example of cryptographic keys for Key-Policy systems.

## 9.8 Using Predicate Based Encryption

Section 9.7 introduced and discussed the two fundamental types of PBE crypto-system and provided an outline of the fundamental steps governing their usage. It was shown that the placement of the access policy will have an affect over how such crypto-systems can be used: it determines access control type. However, policy placement is not the only factor affecting usage. The ownership of the crypto-system, and the number of encrypting and decrypting entities interacting with the crypto-system will also have an effect. From these factors several interesting modes of operation/use can be constructed for PBE crypto-systems. These modes are:

- Public Key Infrastructure (PKI)—where a third-party controls over the Key Authority;
- Multicast Content Provision (MCP)—where an encrypting entity controls the Key Authority; and
- Duty Delegation Infrastructure (DDI)—where a decrypting entity control the Key Authority.

These three modes of operation describe the different ways in which PBE crypto-systems can be used. In this final section of the chapter, these modes of operation are introduced and elaborated upon. First the factors affecting use are introduced:

**Ownership** The first factor discussed is that of crypto-system ownership. Recall from Section 9.2 that ownership of a PBE crypto-system is determined by the owner of the KA. This owner can be one of three types of entities: encrypting; decrypting; or neither, *aka* a third party. The different owners can be said to have different priorities based upon their role within PBEs' ecosystem i.e. encryption. It is this role that will affect the purpose of the crypto-system. Boneh and Franklin [BF01, Section 2] also discusses the affect that ownership had when using IBE.

**Number of Users** With any crypto-system there will either be one or many, encrypting *or* decrypting entities, which when permuted offer the following possible combinations: 1) One encrypting, One decrypting; 2) Many encrypting, Many decrypting; 3) Many encrypting, One decrypting; and 4) One encrypting, Many decrypting. The *one encrypting, one decrypting* relation is not in itself useful and does not leverage the inherent functionality offered by PBE crypto-systems. The remaining scenarios present some interesting scenarios that allude to the ways in which PBE crypto-systems can be used.

**Predicate Placement** The effect that predicate placement has on PBE crypto-systems was described in Section 9.7. Ciphertext-Policy provides a permissive user-centric ABAC model in which access is determined by the attributes the decrypting entity possesses. Ciphertext-Policy provides a proscriptive data-centric ABAC model in which access is determined for the decrypting entity by the KA. Although the use of either type of scheme can be permitted, a schemes suitability per mode of operation will nonetheless be affected by the access control provided.

### 9.8.1 Third-Party Owner: Public Key Infrastructure Mode

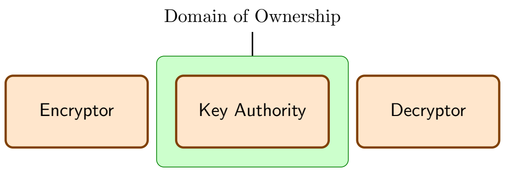


Figure 9.6: Third Party Owner

Throughout this chapter the KA has been presumed to be an independent third-party, that is the interest of the KA is solely that of key provision and management. This set-up represents the 'standard use case' for PBE. The KA facilitates the cryptography as used by the encrypting and decrypting entities: *the cryptography is offered as a service.* In essence, the third-party owner is providing a PKI in the traditional sense. Other entities use the cryptosystem to encrypt data under the *proviso* that entitlement to decryption keys has been validated by the third-party. This 'generic' mode of operation can thus be referred to, simply, as the Public Key Infrastructure mode of operation. Taking into account the number of encrypting and decrypting entities it is clear that combination of many decrypting and many encrypting entities provides the only mode of operation. The other combinations are just specific instances that can easily be represented within the context of many encrypting and many decrypting entities.

With a third-party being the KA it implies that the third-party will not have a vested interest in the data that is to be encrypted. Rather, the 'vested interest' is in the entity's themselves. Given Ciphertext-Policy scheme's disposition towards the provision of user-centric access control, Ciphertext-Policy schemes are more preferable to use as the underlying PBE scheme. Moreover, if a Key-Policy scheme were to be used then this would imply that not only would the third-party owner have some say in access policy creation but also that access control is defined per decrypting entity.

### 9.8.2 Encrypting Entity as Owner: Multicast Content Provision Mode

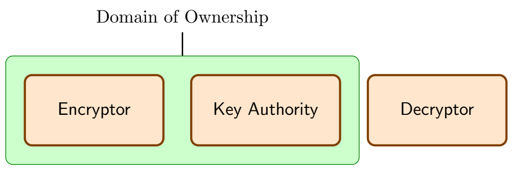


Figure 9.7: Encryptor as Owner

With an encrypting entity in control over the KA, the owner is encrypting their own data and is assigning the ability to decrypt the data to other entities. Here, the encrypting entity should be viewed as the *distributor* of their own content. PBE is used to mitigate access to the content to select entities i.e. multicast access, through the assignment of decryption keys—cf. *Targeted Broadcast* Goyal, Sahai et al. [GS+06]. This mode of operation can be referred to as: MCP. This mode of operation encompasses several different ways in which ones content can be provided: the content stream and the content pool. The stream prescribes the active distribution of content. The pool is when the content has been collected together i.e. a database, and stored together. With these viewpoints, and mode of operation, to only have a single decrypting entity leaves no obvious benefits or usefulness. This leaves only situations in which the number of encrypting and decrypting entities involved is: one, many, and many, many, respectively. The only difference being is that with the latter, the owner has given the ability to encrypt, and thus add content for distribution, to other entities as well as itself.

With the provision of a stream/pool of content this mode of operation is inherently data-oriented. Ciphertext-Policy schemes are suited in this situation if the access afforded to a decrypting entity is to be permissive, or based upon who the entity is. However, the use of Key-Policy schemes is more suitable, not only do they provided proscriptive access control but that the decryption key assigned to a decrypting entity imitates, albeit crudely, a search query. Here the use of KP-PBE is analogous to key word based encrypted search [BDC+04]. Regardless of the underlying scheme this mode of operation affords the owner complete control over key management and to whom decryption of their content is permissible.

### 9.8.3 Decrypting Entity as Owner: Duty Delegation Infrastructure Mode

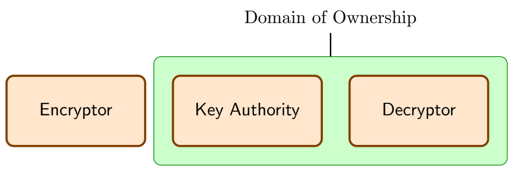


Figure 9.8: Decryptor as Owner

A decrypting entity in control over the KA implies that PBE will be used to mitigate access to data that has been encrypted for the owner by other entities. PBE allows for the owner to delegate access the cipher-text to other, select, entities. The *raison d'être* for this mode of operation is to facilitate the delegation of decryption *duties* to other entities. As such this mode of operation can be referred to as a DDI. This mode of operation is similar to the MCP mode in that the aim is to restrict access to content to other entities, however, the flow of information is inward rather than outward. Within this setting a single encrypting entity or a single decrypting entity provides no obvious benefits, only many encrypting and many decrypting entities.

As with the MCP mode of operation, a more proscriptive form of access control is preferable i.e. Key-Policy schemes. The emphasis is on restricting explicitly the cipher-texts that a decrypting entity (aside from the owner) can decrypt. The owner already allows the decrypting entity the ability to read the owners' messages but the trick is to limit which messages the entity can read. A permissive form of access control i.e. Ciphertext-Policy, does not allow for this. Moreover, with Ciphertext-Policy schemes the access policy is decided, ultimately, by the encrypting entity rather than the Key Authority.

Chapter 10

# Evaluating Predicate Based Encryption

*Predicate Based Encryption schemes are evaluated discussing their novelty, benefits over existing schemes, security issues and deployment worries.*

## 10.1 Novelty of Predicate Based Encryption

The novelty of PBE schemes is derived from: a) the use of attributes to define cryptographic keys; b) the loose link between encryption and decryption keys; and c) the novelty of the construction of PBE schemes. Unlike existing asymmetric cryptographic schemes where the encrypting and decrypting keys can, on their own, be seen as 'numbers'. The set of attributes assigned to an entity are not just 'numbers'. They represent attributes that are intrinsically linked to the entity, data or environment in which the scheme is being used. This link stems from the attributes themselves being mapped to the group elements used in the underlying cryptography. One of the more interesting aspects of PBE schemes relates to the creation of the cryptographic keys. In traditional asymmetric schemes the encryption key is generated, and is linked to, the decryption key. Within PBE the cryptographic keys are not paired and are created independently from each other. Decryption keys are generated by the Key Authority and encryption keys by the encrypting entity. This gives rise to an temporal aspect to the relationship between entities and keys. Data can be encrypted for an entity before their decryption key has been created. The final novel aspect of PBE schemes stems from their construction. Recall from Section 8.3 that SSSs are used to control access over the secret exponent used during encryption. It is through the use of SSSs that the *loose link* between cipher-texts and decryption keys can be achieved. The application of a secret sharing scheme to 'distribute' the secret value used to encrypt the plain-text data, and then specify this distribution according to some predicates is a rather insightful combination.

## 10.2 Comparison with Existing Systems

Section 7.1 introduced the idea that with the use of PBE schemes comes selective fine-grained access control over access to encrypted data. If this functionality was to be compared to existing asymmetric and symmetric crypto-systems in terms of communication styles then: symmetric systems can be seen as providing *broadcast* access control, access is targeted for everyone; traditional asymmetric systems can be viewed as providing *unicast* access control, access is targeted for a particular entity; and PBE systems provide *multicast* access control, access is targeted for a particular set of entities. This 'multicast' communication style exhibited by PBE systems can also be achieved using existing techniques by using: a) only *Asymmetric* cryptography; or b) a *Combined Scheme* that uses a combination of both asymmetric and symmetric schemes for operation. This section illustrates, and discusses the differences between, how existing systems can be used to replicate the functionality seen when using PBE scheme (Section 10.2.1) and also compare to PBE schemes in terms of: a) Number of Cryptographic Operations—Section 10.2.2; b) Key Management Overhead—Section 10.2.3; and c) Cipher-text Management—Section 10.2.4.

### 10.2.1 Replicating Functionality of PBE

#### Using Only Asymmetric Encryption

When using only asymmetric encryption, the encrypting entity needs to obtain the public encryption keys for each intended recipient. Using these keys the entity, say *Albert*, encrypts the message $M$ and sends the resulting cipher-text through some medium. The recipients retrieve $M$ using their decryption keys. Informally, the core steps of this operation are:

```
1. for Entity $e$ in Recipients do
2.     $Albert \rightarrow e : \{|M|\}_{\mathsf{Enc}(r)}$
3.     // On receipt of the message each recipient will do
4.     $M := \{|\{|M|\}_{\mathsf{Enc}(e)}|\}_{\mathsf{Dec}(e)}$
5. end for
```

When using only asymmetric encryption $M$ is encrypted individually, using different keys, for each recipient. This will involve $n$ encryption operations, where $n$ is the number of recipients. Moreover, asymmetric schemes are known to be computationally expensive. A combined approach reduces this computational cost.

#### Using a Combined Approach

The previous implementation can be improved upon with the encrypting entity first encrypting the message using a symmetric scheme and then encrypting the symmetric key using asymmetric encryption *à la PGP*. The resulting cipher-text and encrypted, using the recipients public key, symmetric key will then be sent to each intended recipient. To obtain the message the recipient uses their decryption keys to extract the symmetric key and then use that key to decrypt the cipher-text. Informally the core steps of the protocol are as follows:

```
1. let $CT := \{M\}_{GroupKey}$
2. for Entity $e$ in Recipients do
3.     $CT_{groupkey} := \{|GroupKey|\}_{\mathsf{Enc}(e)}$
4.     $Albert \rightarrow e : CT, CT_{groupkey}$
5.     // On receipt of the message each student will do
6.     $GroupKey := \{|CT_{groupkey}|\}_{\mathsf{Dec}(e)}$
7.     $M := \{CT\}_{GroupKey}$
8. end for
```

Through use of the combined approach the number of encryption operations will be increased to $n + 1$. Furthermore, for each recipient the *GroupKey* needs to be encrypted differently on a per entity basis.

### 10.2.2 Comparing Number of Cryptographic Operations

Comparing the number of cryptographic operations required i.e. encryption and decryption, both the PBE based system require the least number of operations as performed per entity: one decryption and one encryption. On the other hand, with asymmetric and combined solutions the number of encryption operations grows linearly with the number of recipients.

The constant size for PBE was expected, access is specified within the access policy and not through the use of individual public keys. However, the number of cryptographic operations bares little reflection upon the computational cost associated with each operation. The computational cost, which is dependent upon the construction of each scheme, could differ substantially between each type of solution. It is well known that asymmetric crypto-systems are computationally more expensive when compared to symmetric crypto-systems (hence the existence of the combined scheme) and pairing based encryption schemes even more so.

### 10.2.3 Key Management Differences

The operational overhead resulting from key management is significantly reduced when using PBE systems. When the encrypting entity needs to reissue/issue a group key in the combined approach the new key needs to be encrypted and propagated to all intended recipients. Furthermore, this implies that the entire set of intended recipients is known *a priori*. Regardless, when using the traditional schemes it implies that the encrypting entity needs to actively maintain a list of intended recipients. When using PBE not only does the number of encryption operations remain constant, at one but the encrypting entity does not need to maintain a list of intended recipients. This is done implicitly due to the novelty of PBE schemes. However, with PBE schemes other additional key management issues can arise. This is discussed further in Section 10.5.5. Furthermore, as it stands there are no means through which key revocation can be performed explicitly; attributes have validity periods, while with traditional asymmetric schemes the keys are revocable directly.

### 10.2.4 Cipher-text Management Differences

With the asymmetric schemes multiple copies of the plain-text, encrypted with different encryption keys, exist: one copy per receiving entity. This increases the number of cipher-texts that contain the same message that can then be used in *known-plaintext* attacks by a recipient against other recipients, or in a cipher-text-only attack by others. Similarly with the combined approach although there was a single copy of the encrypted data, multiple copies of the group key did exist. Thus, allowing attackers the ability to conduct ciphertext-only attacks on the group key. With PBE crypto-systems only one cipher-text is created and reduces the risk of such attacks.

## 10.3 Mode of Operations

The three mode of operations introduced in Section 9.8 each present a specific way in which a PBE scheme can be used. MCP presents a situation in which a single, or multiple authors, can restrict access to the content they produce. The DDI mode of operation allows for submitted content to be searched by the receiving entity but also by other authorised entities. The final mode, PKI mode, represents a more generic and traditional mode of operation, allowing entities to communicate securely with each other. With the generality seen with the PKI mode similar functionality as seen with the DDI and MCP modes of operation can be achieved. However, this would require that the data sender (for MCP) and data receiver (DDI) would have to involve and thus trust a third party in relation to any form of key management. This could lead to the encrypting/decrypting entity unknowingly allowing unauthorised parties (to them) access to their data. When compared to the other modes of operation the data sender and receiver have complete control over key management and the assurances therein, a third party need not be used.

## 10.4 Security Issues

This section presents several security issues that arise from the use of PBE schemes as part of a crypto-system. For a discussion towards the mathematical, and construction related, security issues of PBE schemes readers are asked to consult Section 8.6.

### 10.4.1 Attribute Revocation

An interesting dilemma with PBE schemes is the issue of key revocation. Section 9.6 already discussed a popular means through which a form of key revocation can be achieved. However, with these schemes the instant revocation of cryptographic keys is difficult. As keys will be issued using attributes that have an expiration date, there is a window between when the KA decides to revoke an decryption key (or encryption key) and when the attributes naturally expires. As a result, newly unauthorised entities can still decrypt messages. There is a trade-off between how long the KA wishes to allow an attribute to be *active* i.e. its validity period, and the risk involved of use by an unauthorised entity.

### 10.4.2 System Integrity

The use of PBE within a crypto-system relies upon several components to work together. Compromising any one of this components can affect the operational integrity of the PBE scheme. Moreover, multiple decrypting entities may collude in an attempt to decrypt messages that own their cannot do so. In this section these issues are discussed further.

**Key Authority Integrity** The KA itself is comprised of several different components that need not reside or operate under the same administrative domain. These components may well be administered and operated independently of each other. If one of these components were to be compromised the integrity of the Key Authority will no longer be intact.

- *Decryption Authority and Encryption Authority* The effects of compromising the DecAuth and EncAuth are similar. A malicious entity will be able to circumvent, or unduly influence, any identification and assignment checks performed when issuing keys. As such cryptographic keys can be constructed and issued, incorrectly and moreover falsely. Furthermore, the key requests of legitimate entities may also be ignored.

- *Decryption Key Generator.* If a malicious entity were to compromise the DKG then the entity will be in a position to construct any decryption key they so wish. This has the untoward affect of allowing malicious entities to then infringe upon the confidentiality of encrypted data that can be decrypted using the keys.

- *Attribute Authority* Compromising the AA allows a malicious entity to influence the composition of the attribute universe $\mathcal{U}$. Allowing for $\mathcal{U}$, and thus the system, to be populated with falsely defined attributes.

- *Revocation Authority* By compromising the RA, a malicious entity can affect the revocation details of attributes and as a result access policies. Allowing for attributes to be revoked prematurely or not at all.

- *Escrow Authority* Compromising the EA gives the malicious entity access to all the decryption keys stored. These keys can then be used to impugn upon the confidentiality of encrypted data.

**Single Point of Failure** PBE schemes are susceptible to *Denial of Service* attacks, if the Key Authority were to be made unavailable then entities would not be in a position to acquire/update their cryptographic keys. Furthermore, the operational integrity of any crypto-system that uses PBE is at risk if any of the internal components of the Key Authority were also to be made unavailable.

**User Collusion** With PBE schemes there is a risk of user collusion, multiple decrypting entities can attempt to combine their decryption keys in order to decrypt some cipher-text. As discussed Section 8.6.2, PBE schemes are collusion resistant. Decryption keys contain a random value that affects the use of the key during decryption. It is this random value that prevents two or more keys from being combined.

### 10.4.3 A Reputable Key Authority?

So far the maliciousness of the KA has been assumed to be false, the KA partakes not in any fraudulent nor malicious activity. With the different modes of operation discussed in Section 9.8 the problem of a malicious KA is only prevalent when a third-party is owner: the PKI mode of operation. The remaining two modes alleviate the problem of a malicious KA to a fair extent. The owner/operator of the KA has s vested interest in its correct operation. For example with the MCP mode of operation the aim is for the crypto-system owner to restrict access to the content that it encrypts itself. To be fraudulent with respect to this access seems to be an unfounded position for the owner to take.

With a third-party owner, entities are reliant on this owner to perform their duties dutifully and more importantly correctly. Suppose that the owner is malicious. The owner will be able to purposefully assign fraudulent decryption keys to other entities or themselves. When using non-attribute hiding schemes (see Section 7.3.3) a malicious KA owner can assign themselves a decryption key that will be able to decrypt a cipher-text upon inspection of the cipher-text and its *meta-data*. With non-attribute hiding schemes extra guarantees/measures are needed from the crypto-system owner to not commit fraud. These guarantees/measures may be technical or legal based and are outwith the scope of this discussion.

### 10.4.4 Privacy

Unless the crypto-system has been constructed using an Attribute Hiding scheme, by default the access policy/attributes associated with the cipher-text will be publicly available. They are embedded along side the actual encrypted value of the plain-text data and ass such are accessible—see Algorithm 3. This functionality, in some cases, could be detrimental to security afforded by the use of PBE. The attributes and access policies represent a hidden information channel that can be abused—see Section 3.6.2.

The need for Attribute Hiding schemes is also dependent upon the setting in which the system is to be used used. For instance, consider the setting of a company that uses a Ciphertext-Policy scheme. It is quite plausible for the attributes assigned to entities to consist of *well known* attributes e.g. department and office number. It can be argued that the access policies, as well as the attributes involved, need not be private; well known information has been used. Securing the payload provides good enough security when compared to hiding the value of the access policy. Moreover correct operation of the PBE scheme ensures end-to-end confidentiality of the message contents.

On the other hand within a military setting the value of the attributes themselves and the access policy would require Attribute Hiding. The attributes themselves can allude to sensitive meta-data such as clearance level i.e. top secret. Disclosure of these values as well as the access policy itself unintentionally leaks information about the encrypted data. Allowing attackers a means to determine the *worth* of the cipher-text, that is attackers will be able to distinguish between cipher-texts accessible to those with low-level clearance and those that require a higher clearance level. Moreover, attackers will be able to use this information to obtain decryption keys fraudulently or even encrypt information.

## 10.5 Deployment Issues

This evaluation concerning PBE ends with a discussion concerning several deployment issues that can arise when deploying PBE schemes. These issues relate not to the security issues (see Section 10.4) but to those concerned with management and usability of PBE schemes.

### 10.5.1 User Interaction

While the KA is responsible for the construction of decryption keys, the responsibility for the construction of encryption keys lies with the encrypting entity. However, this implies that the end user will have to be able to construct suitable canonical attributes for use within an encryption key. For Ciphertext-Policy schemes this problem is exacerbated further with the additional requirement that encrypting entities are also responsible for the construction of correct policy rules and their conversion into access policies. There is a need when deploying and using PBE schemes that a suitable interface can be constructed that abstracts over the complexities arising from specifying attributes or access policies.

### 10.5.2 Computational Cost

Pairing-based cryptography is known to be computationally expensive. The greater the number of pairing operations within a PBE scheme will increase the time required to perform the encryption and decryption step, similar cost is seen when using large plain-text data. The cost associated through large plain-text data can be abated through the use of a combined scheme in which the message is encrypted using symmetric cryptography and the symmetric key is encrypted using PBE. Taking the example setting used in Section 10.2, the steps associated with this combined approach would be as follows:

```
1. let $CT := \{M\}_{GroupKey}$
2. let $CT_{groupkey} := \mathsf{Encrypt}(\mathsf{MPK}, GroupKey, \mathsf{Enc}(\text{Some PBE Key}))$
3. for Entity $e$ in Recipients do
4.         $Albert \rightarrow e : CT, CT_{groupkey}$
5.         // On receipt of the message each student will do
6.         $GroupKey := \mathsf{Decrypt}(\mathsf{MPK}, \mathsf{Dec}(e), CT_{groupkey})$
7.         $M := \{CT\}_{GroupKey}$
8. end for
```

### 10.5.3 Cost of Numerical Attributes

Numerical attributes can be implemented on top of PBE. Section 7.6.3 already described how they can be implemented as 'miniature access-policies'. However, the addition of numerical attributes and the operations performed on them increases the complexity of the access policies and set of attributes used as decryption keys. For each $n - bit$ numerical attribute $n$ attributes are required for representation. Using $64 - bit$ integers, each decryption key in Figure 9.3 will use $64 \times 6 = 384$ attributes rather than the six actually depicted. Similarly, for access policies, each comparison operation will require $\mathcal{O}(n)$ gates. Again using $64 - bit$ integers, each decryption in Figure 9.5 will require the use of $64 \times 3 = 192$ additional gates, one per comparison. Again this contrasts to the simplistic view of only seeing five operations within the access policy.

### 10.5.4 Sourcing Attributes

It is without question that attributes are an integral component of PBE. When discussing the deployment of PBE schemes, there are several issues that need to be addressed: a) the origin of the attribute universe; b) assignment of cryptographic keys; and c) how attributes can be used. From Chapter 9 several points concerned with the attribute universe and key management were established. Firstly, the choice of attributes and their semantic meaning is at the discretion of both the Key Authority and encrypting entities, with the Key Authority being primarily responsible for the attribute universe $\mathcal{U}$. Secondly, the Key Authority is also responsible for verifying an entity's right to be assigned cryptographic keys and also the selection of the used attributes. Thirdly, the attribute semantics is not only influenced by the setting in which scheme will be used but also by the underlying type of PBE system. It is the latter two points that are of interest and can be addressed using existing technologies.

**Identity-Based Attributes** With Ciphertext-Policy schemes the access control implies the use of attributes to describe an entity's identity prescribing the need for some form of identity management to define the attributes and an entity's right to them. While, the KA can themselves be responsible for identity management it can also be performed by other parties by identity providers. Examples of existing identity providers that could be used include: *OpenID* [OpenID], *Windows CardSpace* [CardSpace] or *Higgins Project* [HigginsProject]. However, identity management, and the security and usability issues therein, is a contentious issue [AHS11] that lies outwith the scope of this thesis. Moreover, it can be the case that the provided attributes only provide attributes covering a subset of an entity's identity, more attributes may be needed that can be sourced manually. Within the confines of an organisational setting, for example, the addition of attributes, and ownership verification, can be performed through official mediums/occasions. For example within a University, extra information such as matriculation number, enrolled courses and degree program can be discerned during yearly matriculation and programme/course induction/enrolment.

**Environment and Resource-Based Attributes** Sourcing and verifying ownership/right to possess for attributes used to describe an environment and resource is more troublesome than an entity's identity. The problem relates to the provenance of the attributes and the veracity of claims unto the attributes. For time related attributes this will imply the use of a trusted time stamp authority. For attributes that refer to the encrypted data or some other (in)tangible aspect of the data, sourcing the attributes and to whom they can be assigned is not trivial. Fortunately, some attributes are more predictable that others and as such can be known *a priori* to their initial use.

### 10.5.5 Policy Administration

The policy model described in Section 9.5.1 allows those working with access policies to reason over the construction of access policies, in an abstract manner. However, this adds additional overhead to those entities responsible for the construction of access policies.

**Key-Policy Schemes** For Key-Policy schemes the burden of responsibilities lies with the KA and as such the decrypting entity need not be aware of the complexities surrounding the construction of their decryption key. They need *only* to request a decryption key and they get one in return.

**Ciphertext-Policy Schemes** Within Ciphertext-Policy schemes the burden of responsibility lies with the encrypting entity, unlike Key-Policy schemes this burden is greater. Encrypting entities have to manage access policies, the complexity of which is determined by whom the encrypting entities wishes to encrypt their data for. However, as mentioned in Section 10.5.1, a good interface will aide in reducing the complexity of constructing policy rules and access policies.

### 10.5.6 Key Storage

Although, the issue of key management was discussed in Chapter 9 a concern when deploying PBE schemes is that of key storage. Both encryption and decryption keys will need to be stored by an end user. This will include expired decryption keys used to decrypt messages accessible only by those keys. When storing decryption keys this will imply storing a decryption per validity period of the attribute. In resource constrained environments the issue of decryption key storage might pose a problem.

**Key-Policy Schemes** For Key-Policy schemes, encryption keys are comprised from a set of attributes. Managing these attributes and their storage is negligible, individual attributes can be stored efficiently as *plain* strings. However for decryption keys, comprised from access policies the, size of a single key will be $n \times$ bit-length of a group element, where $n$ is the number of leaf nodes in the access policy—see Section 8.6.3. As the complexity of the access policy grows, so will its size. This is especially pertinent for the use of numerical attributes and numerical comparisons that have a hidden underlying cost—see Section 10.5.3.

**Ciphertext-Policy Schemes** As with encryption keys for Key-Policy schemes, the storage requirements for policy rules and any generated access policies can be abated through their storage as plain strings. As with decryption keys for a set $\mathcal{S}$ the key generated will be $\mid \mathcal{S} \mid$ $\times$bit-length of a group element in size. As the number of attributes in $\mathcal{S}$ grows so to will its size. As with KP schemes this is especially pertinent for the use of numerical attributes and numerical comparisons that have a hidden underlying cost.

# Part III

# Leveraging Predicate Based Encryption in the Cloud

Chapter **11**

# Scenarios for using PBE in the Cloud

*Five scenarios for using Predicate Based Encryption within the Cloud are presented. These scenarios differ in terms of: mode of operation, predicate placement, and ownership over the Key Authority.*

## 11.1 Overview

The focus of this thesis now turns towards how Predicate Based Encryption (PBE) can be used within the cloud. Part II introduced PBE and discussed how it provides for an efficient and secure single-encryptor/multi-decryptor environment. Section 9.8 introduced three modes of operation that characterises how a PBE scheme can be used within a crypto-system. The Public Key Infrastructure (PKI) mode allows a third-party to offer other entities a means through which they (the other entities) can share their data. The Multicast Content Provision (MCP) mode allows an entity the means through which they can selectively share their data with others without having to rely upon a third-party. The final mode, that of Duty Delegation Infrastructure (DDI), allows an entity to restrict access to data that has been sent to them. However, the application of these three modes of operation to a Cloud setting will be affected by: a) whether the Key Authority (KA)—the owner of the PBE scheme—is a service user or a service provider—Cloud Service Provider (CSP); and b) the style of access control required i.e. Ciphertext-Policy (CP) or Key-Policy (KP). From these factors, five viable scenarios emerged:

**Scenario** I: *Embedded within a Service.* A CSP incorporates a CP-PBE scheme in PKI mode within an existing service. Providing service users the means with which they can share data amongst each other within the domain of a particular service.

**Scenario** II: *PBE-as-a-Service.* Similar to Scenario I with the exception that PBE is used outwith the confines of a particular service domain.

**Scenario** III: *'Database' Submission.* A CSP offers a service in which service users can submit data to a database. A KP-PBE scheme in DDI mode is used to restrict access over the submitted content to authorised employees of the CSP and other affiliated parties.

**Scenario** IV: *'Database' Access.* The antithesis of Scenario III. A CSP offers content that can only be accessed by subscribed users. A KP-PBE scheme is used in MCP mode to restrict subscribed users' access to the CSP's content.

**Scenario** V: *'Distributed Security'.* The *service user* deploys their own CP-PBE scheme in MCP mode. PBE is used to mitigate access over data the user pushes to the Cloud.

This chapter introduces these different scenarios and what they pertain to. Within Chapter 12 further discussion over these scenarios and other issues can be found.

## 11.2 Leveraging PKI Mode

The PKI mode of operation leans naturally towards the *as-a-Service* paradigm, and consequently its use by a CSP. The CSP *is* the third-party offering the use of PBE to its service users. With this in mind, PBE can be offered by the CSP in two distinct ways, either: a) embedded *within a service*; or b) provided *as a service*. In both cases, the CSP is promoting the sharing of data amongst other users of the services; sharing is a dominate feature exhibited by many Software as a Service (SaaS) services. Leading to the implication that these two scenarios, at least, appear to be best suited for SaaS services. Moreover, an access control style is required that allows the encrypting entity to state explicitly for whom access will be granted on a per message basis. As was discussed in Section 9.8.1, Ciphertext-Policy schemes provide such access control. Indicating that the attribute universe is used to, at least, describe users of the service.

### 11.2.1 Scenario I: Embedded within a Service

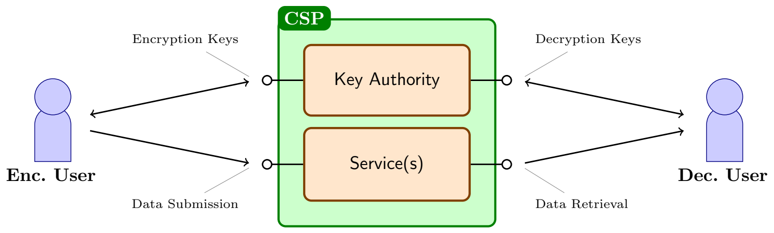


Figure 11.1: Outline of Scenario I

Embedding a CP-PBE scheme within an existing service allows for the complete integration of PBE within the service itself. Figure 11.1 outlines the core interactions within this scenario. Upon registering with the service, new users are provided with a decryption key derived from a set of attributes that describe the user. When pushing data to the service, service users encrypt their data under access policies of their choosing. Moreover, as the KA is embedded within the service, the CSP can also aid in policy administration. In other words, the CSP can offer service users a means through which to specify policy rules and thus construct access policies.

Due to embedding PBE within the service itself, the attribute universe can be extended to refer to service specific functionality such as the CSP's ability to read the users data for targeted ads. For example, take an Online Social Network (OSN). The attribute universe can identify service users through attributes such as: identification number, gender, school networks, interests, location, D.O.B. *et cetera*. Functionality such as, the visibility of users' messages can also be included within the same attribute universe. Following on from indicating the visibility of a user's message is the notion that the CSP can also be referenced within the policy rules and treated as a user in their own right. Though this does not adhere strictly to the PKI's *modus operandi* (see Section 9.8.1) it nonetheless allows the encrypting user to explicitly opt-in the CSP when the user constructs policy rules. Moreover, such an attribute can be combined with a date attribute to provide the CSP with a limited window of opportunity to access encrypted data. This is a powerful construct, the user has explicitly stated within an access policy that the CSP can access the encrypted data, and more importantly for how long.

A problem for this scenario is that service users need to trust the CSP. As the CSP is the KA, the CSP could easily construct their own arbitrary decryption keys, and use these keys to decrypt service users' messages. Service users still need to trust the CSP to not be malicious in this respect. As discussed in Section 3.9.1, this trust can be enforced through legal means i.e. privacy policies and service level agreements. Furthermore, using PBE as described requires that each CSP provide their own scheme. With each service that the user interacts with a different PBE scheme needs to be taken into account. That is, separate key management facilities and algorithms need to be managed by the user; one per service. Given what is known over key storage issues (see Section 10.5.6) this could be a problem. Although, how the PBE is actually deployed will affect this somewhat, see Section 12.3 for more information.

### 11.2.2 Scenario II: PBE-as-a-Service

Providing a CP-PBE scheme as a service in itself allows service users to interact with multiple services, offered by different CSPs, and use a common solution. This reduces the overhead concerning key management, and also the different encryption and decryption algorithms that a service user must be aware of. Providing 'PBE-as-a-Service' makes this scenario an example of

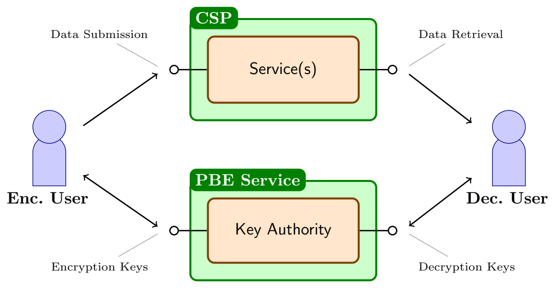


Figure 11.2: Outline of Scenario II

*Security-as-a-Service*—more specifically it can be tied directly into *Identity-as-a-Service* services. This implies that not only can this scenario be used to protect SaaS level services but also at the Platform as a Service (PaaS) level as a service. Figure 11.2 outlines the core interactions within this scenario.

Like Scenario I, users possess decryption keys comprised of a set of attributes used to describe the user themselves. When interacting with different services the user can push encrypted data knowing that only others who can satisfy the access policy can access the underlying plain-text data. Policy administration service can also be offerd by the PBE service. As with Scenario I CSPs should be treated as users in their own right. Hence, when constructing policy rules the encrypting user can explicitly opt-in the CSPs as someone with whom they wish to share their data with. Moreover, CSPs 'service users' can themselves integrate this pre-existing PBE service into their own service offerings. Unlike Scenario I, however, service users can treat the CSP like any other user and not have to worry about the CSP's maliciousness. A CSP in this setting is unable to construct arbitrary decryption keys. Users can use a service even though they may have no trust in the CSP offering that service. When the user does have trust in the CSP to access the user's data, then the CSP can be opted in.

However, the service agnosticism does present several interesting issues. For one, the attribute universe is defined by a third-party who may not possess knowledge of service specific information, offering users attributes that describe information outwith the context of a service. This may affect the expressiveness of policy rules as defined by encrypting users. Offering a PBE service does reduce the overhead in terms of key management and knowledge of algorithms, however, this reduction in overhead comes at some cost. Scenario I is highly compartmentalised, CSPs are only able to access the data that has been pushed to their service. With the solution described in this section the PBE service provider, if malicious, has the ability to construct decryption keys to access data that a user has pushed to multiple services.

**Remark.** *A recurring element with Scenarios I and II is that the KA is a third-party distinct from the service user. While this does simplify the involvement of the service user it requires the service user to be reliant upon the KA to adhere dutiful to its remit—see Section 10.4.3. Such misgivings can be addressed with the MCP mode of operation that removes the need for a third-party KA.*

## 11.3 Leveraging DDI Mode—Scenario III: 'Database Submission'

The rationale behind the DDI mode is controlling access over submitted data. When a service user uses this mode of operation it implies that the service user themselves will be controlling access to data that has been submitted to them: This has no obvious benefits or uses. It is better for the user to utilise the MCP mode of operation—see Section 11.4. On the other hand with a CSP as the KA, an interesting use case does emerge.

With DDI mode the CSP is using PBE to facilitate selective access over data, submitted expressly for use by the CSP, under the proviso that said data will only be accessible by the CSP's own employees and affiliates. That is data is submitted by service users and the CSP allows authorised entities access to select subsets of *all* data that has been submitted. This essentially describes the operation of a database. Databases can be thought of in terms of: a) who is submitting the data; b) who is accessing the data; and c) the data itself. With DDI mode, the database is stored with the CSP. Data is submitted *for* the CSP by service users, and is being accessed by the CSP's employees and other affiliates. Figure IV outlines the core interactions for this scenario.

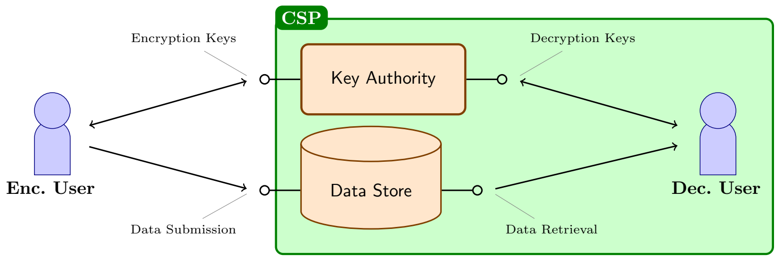


Figure 11.3: Outline of Scenario III

Within databases one needs to control access not only to the database itself but also to the data stored within. Obviously, users' access over the data within the database needs to be proscriptive and users ability to access individual records needs to be curtailed. As discussed in Section 9.7 Key-Policy schemes provide such access control. As such data is encrypted under a set of attributes, and decryption keys from a predicate. Another and more intuitive reason for using Key-Policy schemes stems from the decryption keys. Decryption keys in this setting represent simple search queries. As mentioned in Section 9.8.2 such use of a KP scheme is analogous to keyword based search albeit with a more expressive query mechanism. When submitting data, service users encrypt their data under a set of attributes. The CSP can then internally determine on a per employee basis what the employee can access from this database. Moreover, with numerical attributes it is feasible to limit the period during which the employee can use their query.

The analogy of a 'database' implies the deployment of PBE as part of either a SaaS, or PaaS service. For example, a course submission service could use DDI mode to control submission of students submission. During submission, students could encrypt their course under a set of attributes that describe[1]: a) the student's matriculation number; b) the course code; c) the assignment number; and d) submission time. Lecturers and teaching assistants can then be assigned decryption keys pertaining to the course(s) they are associated with.

**Remark.** *This use of PBE could also be extended to situations in which users of Infrastructure as a Service (IaaS) services require their configuration files to be stored within the cloud.*

## 11.4 Leveraging MCP Mode

Recall from Section 9.8.2 that with the MCP mode of operation an encrypting entity is the KA. Within a cloud setting two scenarios emerge when: a) a service user is the encrypting entity; and when b) a CSP is the encrypting entity. The first scenario provides an alternate solution to those presented in Section 11.2. The second scenario, on the other hand provides a setting in which a CSP can control access to their own content in a manner similar to that seen with the DDI scenario. Although both scenarios require a proscriptive form of access control, the choice of underlying PBE scheme will differ. With a service user Ciphertext-Policy schemes are better suited, access needs to be decided per message. With the CSP as the encrypting entity the setting is more data centric and as such a Key-Policy scheme will suffice.

### 11.4.1 Scenario IV: 'Database Access'

CSPs can utilise MCP mode to restrict access to the content that they produce in much the same manner as was seen with Scenario III—Section 11.3. The difference lies in the involved actors and origin of the data. Figure 11.4 outlines the actors involved and core interactions between them. Employees of the CSP together with authorised affiliates are those permitted to encrypt data. Decryption of data is performed by service users. When signing up to a service, users are assigned a decryption key derived from some policy rule that describes the data they are allowed to access. Such policy rules can be used to indicate service related information such as

[1]Student names have been omitted due to the need for anonymous marking in tertiary level education.

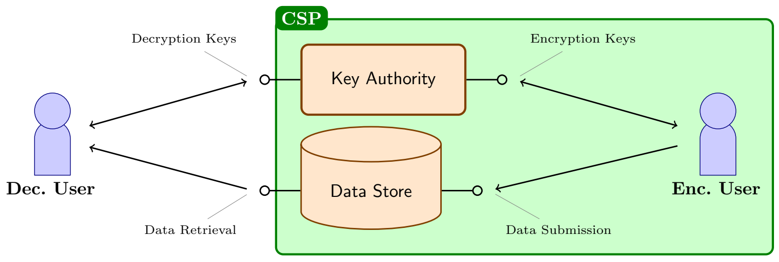


Figure 11.4: Outline of Scenario IV

subscription level and periods of time. This provides the CSP with a means to enforce different subscription models i.e. *freemium*. With MCP mode the act of restricting access can also extend to any content streams that the CSP produces and broadcasts. This implies that this scenario can be used at both the PaaS and SaaS service levels. Take a media streaming service such as Netflicks, streamed content can be encrypted under a set of attributes that describe, for example: a) the content's rating i.e. U, PG, 12, 16, 18; b) the subscription level e.g. free, gold member, silver memeber; and c) the content's details such as title, season and episode.

### 11.4.2 Scenario V: 'Distributed Security'

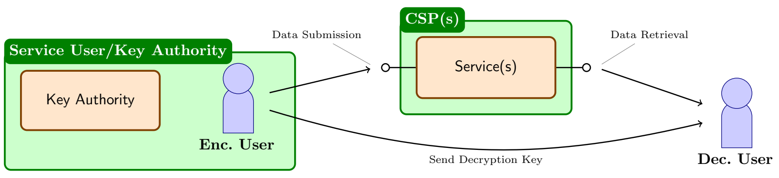


Figure 11.5: Outline of Scenario V

Scenarios I and II present solutions to protecting data they wish to share via the cloud. The underlying problem with these solutions is the reliance upon a third-party to act as the KA. When deployed by a service user the MCP mode presents an improvement with respect to trusting a third-party. With the MCP mode there is no need for a trusted third-party over key management, users provide their own PBE scheme.

When signing up to a service or when starting to interact with a new 'friend', the encrypting user i.e. the Key Authority, will create and assign to the entity a decryption key. This decryption key will be derived from a set of attributes that describes how the decrypting user relates to the encrypting user. Through use of a Ciphertext-Policy scheme, the encrypting user can encrypt their data locally under a policy rule, and then push the cipher-text into the cloud. Attributes can be used to indicate, for example: a) type of relationship with another entity i.e. an associate, a friend, a close friend, a really close friend, family, or CSP; b) visibility of message i.e. is the message to be private or public; and c) an identifier for each domain that the user has interacted with i.e. on a OSN.

This use of the MCP mode of operation presents a decentralised solution to the problems associated with Scenarios I and II. Implying this scenarios use at the PaaS and SaaS service levels, with each 'user' providing their own encryption scheme. From this two interesting observations can be made that can affect the management and deployment of this mode. First, the encrypting user *is* responsible for the remit of the Key Authority. The user is responsible for policy rule administration, responsible for the the correct assignment of decryption keys, responsible for key management *et cetera*. Secondly, the security guarantees provided cover uni-directional communication. Each user utilises PBE to protect their own data only. For $n$ users communicating it requires each user to take into account the decryption keys, and associated decryption algorithms for $n-1$ other PBE schemes.

## 11.5 Summary

Table 11.1 provides a summary of the scenarios presented in this chapter according to: a) mode of operation; b) type of scheme; and c) the entity acting as the KA.

| Scenario | PBE Mode | CP/KP | Identity of 'KA' | Service Level |
|---|---|---|---|---|
| Scenario I | PKI | CP | CSP | SaaS, PaaS |
| Scenario II | PKI | CP | CSP | SaaS, PaaS |
| Scenario III | DDI | KP | CSP | SaaS, PaaS, IaaS |
| Scenario IV | MCP | KP | CSP | SaaS, PaaS |
| Scenario V | MCP | CP | Service User | SaaS, PaaS |

Table 11.1: Summary of the five different scenarios that PBE can be used in the Cloud.

Chapter 12

# Discussing PBE in the Cloud

*The use of Predicate Based Encryption within the cloud is discussed. Data that can be protected is identified. End-to-End security is addressed. Fulfilling the remit of the Key Authority is touched upon, and finally, the five scenarios are compared.*

## 12.1 Overview

Chapter 11 presented five different scenarios that describe how PBE could be leveraged within the cloud. As a denouement to this part, and to this thesis, this chapter evaluates the use of PBE in the cloud. For a discussion over PBE in general and within a crypto-system can be found within Chapter 10. Section 12.2 discusses the data that PBE can be used to protect at each of the three service layers. The effect that placement of the cryptographic operations has over security is discussed in Section 12.3. Section 12.4 discusses how the role of the KA can be fulfilled by both CSPs and service users. Finally, the five different scenarios are compared in terms of the protection offered within Section 12.5.

## 12.2 What can PBE be used to Protect?

Chapter 4 presented two threat models for data in terms of its *life cycle*, from which data is being described as either being: a) in the cloud; or b) outwith the cloud. When pushing and retrieving data from the cloud existing mechanisms (e.g. `HTTPS` and `SSH`) can be used to protect the data *in flight*. However, the problems associated with data in the cloud stem not from when it is *in flight* but rather from once it has become resident within the cloud—see Chapter 6. With PBE, assurances over the confidentiality and accessibility can be made. PBE can aide in preventing the unwanted exposure, unwanted leakage and other unwanted breaches of confidentiality of cloud resident data. Other guarantees such as integrity, non-repudiation and authenticity need additional security mechanisms.

Recall from Section 2.2 that there are three different service levels. The nature of the data that needs to be protected will differ depending upon the exact service layer being addressed:

**Infrastructure as a Service (IaaS)** The IaaS service level is concerned with the deployment of virtual machines. With this service level the data that is of concern are the settings and commands used when configuring, deploying, and managing the virtual machine instances. This is data that will be submitted to the CSP. While commands are to be invoked, settings are to be stored for use.

**Platform as a Service (PaaS)** Services at the PaaS service level are characterised through the APIs exposed by the CSP through some platform tooling. Although, the range of this tooling is vast it can nonetheless be seen as *just* an exposed Application Programming Interface (API) promoting some functionality. The APIs themselves can be characterised as performing some data processing or providing access to some data store.

**Software as a Service (SaaS)** The highest service level, SaaS, deals with services that offer more complex interactions and functionality. The range, shape and form of the data being pushed into the cloud will vary greatly from service to service. For example, the data pushed by the user can include but is not necessarily limited to: messages posted on a social networking site, documents in an online office suite, or photographs posted to an photo sharing site. A dominant feature of many SaaS services is the collaboration, or sharing of this information with other users.

A related discussion to *what* needs to be protected is *when* (and where) the protect be applied. That is where should encryption and decryption of data occur. Such a decision will affect the use of PBE and is discussed within Section 12.3.

## 12.3 Effect of Cryptographic Operation Placement

Chapter 4 presented threat models that describe when data will be vulnerable during its lifecycle. When deploying PBE in the cloud an important question to ask concerns where in the data lifecycle should the encryption and decryption operations be deployed. Such a decision will affect the end-to-end security of encrypting entities' data. At what point during the data lifecycle can the confidentiality of data be assured and when can it be questioned? Moreover, this issue raises concerns over the trusted computing base established by the CSP.

### 12.3.1 Key Authority and Service are not linked

With Scenarios II and V the KA is not tied to any particular service. The deployment of encryption and decryption operations is trivial. Data will be encrypted by the service user prior to its insertion into the cloud—Stage 1. When the decrypting entity is a CSP the data can be decrypted once it has entered the cloud—Stage 2. Finally, service users can decrypt data once it has been received locally—Stage 4. End-to-end confidentiality of the message has been assured.

### 12.3.2 Key Authority and Service are linked

Wwith Scenarios I, III and IV the deployment is not as straightforward. With these scenarios the CSP is the KA and is the provider of the service that service users are interacting with. Generally, speaking two options exist over the placement of the operations:

**Option** 1: Cryptographic operations are performed by service users themselves. Data is encrypted and decrypted at Stages 1 and 4 of the data life-cycle.

**Option** 2: Cryptographic operations are performed by the 'CSP' on behalf of the users. Data is encrypted and decrypted at Stages 2 and 3 of the data life-cycle.

Option 1 places the encrypting user in control of the encryption process. Their data has been encrypted at the first stage within its life-cycle prior to its push to the cloud. This affords the user knowledge of the precise composition of the encryption key used to encrypt the data, and also of the encryption process. Similarly, the decrypting user only accesses the cipher-text and decrypts the cipher-text once it has been received locally i.e. it is outside the domain of the service. By performing the cryptographic operations locally, in the domain of the user, the scope at which threats attacking the confidentiality can occur has been reduced. End-to-end confidentiality has been assured: *The data goes into the cloud encrypted, it leaves the cloud encrypted.*

Option 2 on the other hand, cannot offer the same guarantees. Data pushed into the cloud by the service user has now been encrypted within the domain of the CSP. The entity needs extra assurances as to the key used to encrypt their data. With decryption, the service user's decryption key must now also reside in the service for it to be used. Unauthorised use of this key needs to be assured as well, especially from malicious insiders. Steadfast guarantees over end-to-end confidentiality cannot be assured: *The data sent by the user enters the cloud unencrypted and will leave the cloud unencrypted.*

With Option 2 threats to data, and also to the decryption key can now also occur within the domain of the CSP. Although this option may appear to be a forlorn one it does has its advantages. Section 6.3 made reference to the *Grendel* solution. This solution also placed the cryptographic operations on the service side of the interaction allowing the user to be 'unaware' of the protection given to their data. This not only reduces the perceived complexity but also relieves the service user over the responsibility for maintaining and storing cryptographic keys. This may be of advantage within resource constrained environments, and mobile environments in which the service user does not have continuous access to their decryption keys.

## 12.4 Fulfilling the remit of the Key Authority

Central to the use of PBE as part of a crypto-system is the KA. Chapter 9 introduced the KA together with a discussion over the remit that the KA must adhere to and how parts of this remit could be achieved. This includes responsibility for managing the attribute universe, assigning/issuing cryptographic keys, and for KP schemes policy rule administration. Section 10.5 also discussed how attributes can be sourced, among other deployment issues. However, how a CSP or a service user can fulfil this remit will differ.

**Cloud Service Provider** Naturally, CSPs have access to a comprehensive existing infrastructure. This infrastructure can be leveraged by the CSP when perform their duty as a KA. When including/using PBE as part of a service, the CSP can use existing information from the service to determine the attributes used within $\mathcal{U}$. An existing identity management infrastructure can also be used to verify any claims a user has over attributes. Furthermore, existing policy rule specifications can be used as a base when constructing decryption policies i.e. Key-Policy schemes, or as a guide when users need to construct encryption policies i.e. Ciphertext-Policy schemes. Finally, with Ciphertext-Policy schemes the CSP can also offer an interface, and service, through which users can abstract over the complexities of policy rule specification and management.

**Service User** When a service user is the Key Authority, they will not have the same level of resources and existing infrastructure when compared to a CSP. However, the scale of operations for a service user is vastly different when compared to that of a CSP. Here the service user operates in a smaller more personalised environment. They deal with entities directly i.e. one-on-one, the service user can perform the remit 'manually'. That is, the service user can verify claims over attributes when constructing decryption keys themselves. Unlike a CSP who will have a fixed address, however, the service user is inherently mobile. The service user will be access services from different locations from different machines. How exactly is the service user going to fulfil this remit, if they are continuously on the move?

## 12.5 Comparing Scenarios

The five different scenarios presented in Chapter 11 can be divided into two distinct groupings concerning the style of interaction (between the service user and CSP) that the scenarios are used to protect: bidirectional and unidirectional. This grouping can be used to compare related scenarios. Moreover, Chapter 6 argued that a *good* solution for protecting the confidentiality of data pushed to the cloud should take into account the following three points:

- *User empowerment does not alone equal Data Obfuscation.*
- *The user need not be responsible for ensuring all security guarantees.*
- *Some CSPs may be evil but some are more evil than others.*

### 12.5.1 Comparing Scenarios I, II and V

Scenarios I, II and V describe a means to protect the results of a bidirectional communication style. With this style of communication a service user expects to push their, and pull other service users' data from the cloud. PBE is used to ensure confidentiality of this data.

**User Empowerment** In each of the three scenarios, service users have control over whom precisly can decrypt their data, and by extension how it can be used. They have become empowered. However, the degree of empowerment depends upon the KA. The KA is responsible for defining the attribute universe, and assignment of decryption keys. This in turn affects the expressivness of access policies in terms of what attributes the user can specify, and the effect of application e.g. attributes that identify if the CSP can decrypt encrypted data. Scenario I offers tight integration with a single service. The CSP can design an attribute universe that caters to the needs of the service. On the other hand, Scenario II does not offer as tight an integration. The definition of the attribute universe is dependent upon a third-party who is not necessarily aware of a particular services needs. Finally, Scenario V offers the service user complete freedom over the composition of the attribute universe, and moreover, the service user is also in control of who is able to obtain decryption keys.

**User Responsiblities** With these scenarios, the CSP is not afforded complete trust in their abilities to secure user's data. In Scenario V the service user has *no trust* in the abilities of the CSP to offer data confidentialy. The service user is responsible for all matters, excluding decryption, arising from the use of the PBE scheme. Scenario I offers a solution that allows the service user to have *some trust* in the CSP. The CSP is responsible for all matters excluding the specification of encryption keys, this is left for the service user. Finally, Scenario II implies that the service use has *no trust* in the CSP to protect the confidentiality of users' data. However, with Scenario II this service user must now have some trust in the ability of the 'PBE Service' to fulfill its remit.

**Trust in the CSP** Each of the three scenarios offers differing levels of trust over the CSP. The preceding paragraph explained this in relation to who provides assurances over data confidentiality. Section 11.2 established that a malicious CSP acting as the KA will be able to construct arbitrary decryption keys. This is not a problem for Scenario V. For all three scenarios, however, service users can explicitly stipulate on a per message basis if the CSP can access the service users data.

### 12.5.2 Comparing Scenarios III and IV

Scenarios III and IV, describe a means to protect a distinctly unidirectional style of interaction. With Scenario III, the service user is pushing data to the cloud. With, Scenario IV the service user is pulling CSP's data from the cloud. These scenarios do not necessarily seek to empower service users over the confidentiality of their data: They seek to empower the CSP instead. Moreover, both of these scenarios allow the CSP to provide a form keyword search using expressive queries over an 'encrypted database'.

**Scenario III** Scenario III presents a scenario in which the user is afforded some control over access to their data through specification of the encryption key. Ultimatly, however, it is the CSP that has been empowered. This solution presents users with a solution that they must have *complete trust* in the CSP to control access to the submitted data.

**Scenario IV** With Scenario IV, the three guidelines iterated at the start of this section cannot be applied. This scenario seeks to empower the CSP over access to their data, and not service users. In this scenario the CSP is in control over the attribute universe and over issuing decryption keys. The CSP has guarantees over the confidentiality of their content, and allows them to enforce not only fine grained access control but also support for tiered content provision.

# Part IV

# Conclusion and Further Research

Chapter 13

# Conclusion

Quem deus vult perdere, dementat prius.

Euripides

Cloud Computing embodies the *as-a-Service* paradigm and allows for services to be provided *en masse* to consumers. The problems associated with the use of cloud based services can be summarised by the unknown risk profile (see Section 3.8) and unknown expectation of privacy—see Section 3.9.2. When service users push data to the cloud they need to rely upon Cloud Service Providers (CSPs) adhering to their remit, and doing so dutifully. However, when looking to build solutions to protect data in the cloud it is important to remember that for the service user the CSP *can* be trusted, albeit at arms length—see Section 6.3. The threat models presented in Chapter 4 illustrate that threats to data occur both in the domain of the service user and the domain of the CSP. Traditional privacy models are too user-centric and CSP-fearing when trying to address the problem of protecting data—see Section 6.2. A privacy model centred around Kafka's *The Trial* helps to address this problem, this privacy model indicates that when protecting one's data one should also have control over its use rather than solely preventing its collection: CSPs and service users need to work together.

Predicate Based Encryption (PBE) presents an interesting and also novel family of asymmetric encryption schemes. PBE combines Attribute Based Access Control (ABAC) with asymmetric encryption, allowing for a single-encryptor/multi-decryptor environment to be realised using a single scheme. Replicating such functionality using more traditional techniques requires a more complex approach—see Section 10.2. Even so, PBE is inherently more flexible, data can be encrypted for a decrypting entity prior to the creation of the decrypting entity's decryption key: The precise list of decrypting entities need not be known *a priori*. Such novelty lies with the cryptographic keys—see Section 10.1. Cryptographic keys are not just numbers, there is no one-to-one pairing of encryption and decryption keys, and encryption and decryption keys are created separately from each other.

Though understanding the operation and potential use of PBE schemes was not inherently difficult it was not trivial either. Understanding the means through which PBE schemes can be constructed also proved to be a somewhat tedious affair. Unfortunately with PBE, a disproportionate amount of time was spent searching for, and collating, information from different sources; different papers begat different schemes which in turn begat different terminology. However, these efforts were not without dividends, by learning more about PBE scheme construction, various issues surrounding cryptographic keys were identified and addressed. For instance, blinding values prevent decrypting entities from combining their decryption keys—see Section 8.6.2. The inclusion of numerical attributes also presents an underlying hidden cost (see Section 10.5.3), which together with key revocation techniques will increase the space needed to store decryption keys—see Section 10.5.6. Furthermore, by looking at different schemes a means to categorise the different PBE schemes emerged from such schemes' emergent properties.

Similarly, describing PBE's use as part of a crypto-system was also a necessary evil, it established not only how PBE schemes could be deployed (see Section 9.8) but also points of contention within such deployment. Of which the most notable issues were those surrounding key management such as constructing, issuing and revocation. Furthermore, the *use* of PBE identified three different modes of operation that describe the three different ways in which PBE schemes can be leveraged within an crpyto-system—see Section 9.8. When combined with the cloud setting, two different sets of scenarios emerged based upon whether the service user's or CSP's data was to be protected.

PBE schemes can be used to protect service user's data in three different scenarios: Scenario I saw the inclusion of PBE within a service; Scenario II saw the provision of PBE *as-a-Service*; and Scenario V saw PBE being deployed by the user themselves. In each of these three scenarios PBE can be used by service users to specify precisely with whom they wish to share their data, for what purpose, and for how long. Although Scenario V may be a privacy zealot's ideal choice—they are in full control—its practical feasibility has yet to be determined; the ability for service users' to act competently as a Key Authority is still unclear. The remaining two scenarios, on the other hand, do appear to be more promising. However, these scenarios in themselves do present a dilemma between usability and the guarantees made over end-to-end security—see Section 12.3.

When looking to protect CSP's data, PBE can facilitate keyword search with complex queries over encrypted data: Scenario III by the CSP; and in Scenario IV by a service user. This use of PBE is rather interesting in that the focus of these scenarios is on the CSP and not service user, and is most certainly worthy of further investigation.

The use of PBE within the cloud appears to be concentrated at both the PaaS and SaaS service layers. Though some may be surprised at PBE's lack of use at the IaaS layer, this was not totally unexpected. The primary interaction between a service user and CSP at this level is over managing virtual machines: Not much else happens.

As previously mentioned, comprehending the field of PBE was made more difficult due to the inherently heterogeneous nature of the schemes studied. As such this thesis also sought to provide readers with a decent starting point over PBE schemes and also their use within crypto-systems. However, towards the end of this investigation a plethora of papers have been released addressing PBE in terms of its description and use within a cryptographic system. From Boneh, Sahai and Waters [BSW10], PBE can now be seen under the guise of Functional Based Encryption (FBE), here Boneh, Sahai and Waters provides a coherent, and authoritative, definition towards PBE together with a categorisation of the different FBE schemes. Interestingly the categorisation presented is similar to that presented within this thesis—see Section 7.3. Furthermore, Akinyele, Lehmann et al. [AL+10] utilises PBE to secure medical records, and Bobba, Fatemieh et al. [BF+10] utilises PBE to for secure messaging. Both these solutions show real-life uses of PBE schemes and as part of a cryptographic system. In fact the solutions presented in Bobba, Fatemieh et al. [BF+10] and Akinyele, Lehmann et al. [AL+10] would both correspond to Scenario I.

In summation, the scenarios presented in this thesis do show that the use of PBE in the Cloud is advantageous, moreover recent papers supports these claims. However, this investigation is far from being over. The results here are just the tip of the iceberg, and there are several directions in which this investigation can go in. These new directions are discussed in Chapter 14.

Chapter 14

# Further Research

*Several avenues for further research are discussed. Addressing Predicate Based Signing, Distributed Attribute Based Encryption, Use Case development and practical feasibility.*

Omnium rerum principia parva sunt

Marcus Tullius Cicero

## 14.1 Attribute Based Signing

PBE schemes provide assurances towards the confidentiality of data. They do not provide assurances towards the authenticity of the data. To provide such assurances digital signature schemes are required. Attribute Based Signing (ABS) is an adaptation of Attribute Based Encryption (ABE) for the purposes of providing guarantees towards the *provenance* of the signed data, and moreover towards the *anonymity* of the signer [MPR10]. Within ABS schemes the digital signature indicates that the signer can satisfy the given predicate and hence has attested to the message—*provenance*. Furthermore, the precise set of attributes used by the signer is not revealed—*signer anonymity*. Such a signature scheme has several practical uses. Maji, Prabhakaran and Rsulek [MPR10] discusses the use of ABS schemes for attribute-based authentication, trust negotiation, and the leaking of secrets. Future research would seek to investigate:

- The feasibility of such schemes.
- How such schemes can be leveraged in general: What do they allow for? and What do they bring? and finally
- How these schemes can be used within the Cloud.

## 14.2 Decentralising the Key Authority

The Key Authority is responsible for the management of attributes, attribute definition, and also over attribute assignment to entities. However, these attributes naturally originate from different sources and as such they will also be managed by different authorities. For example, within a university setting students, and information concerning students, are handled by various administrative departments[1]. Matriculation is typically performed by a central administrative body, while more degree programme specific aspects (e.g. advising) are handled at a faculty level, and also further on a per school/department basis. There may not be one central body that collates and is able to verify the attributes/access policies that are to be assigned to the student. The ability for a single Attribute Authority and Decryption Authority to administer over a 'global' attribute universe is made more difficult.

Distributed Attribute Based Encryption (DABE), also known as *Multi-Authority ABE*, is an adaptation of Ciphertext-Policy-ABE that delegates the management of attributes and the creation of decryption keys to other individual attribute authorities [LC+10; MKE09; Cha07]. Future work will be to investigate:

- The feasibility of such schemes.

[1] This will no doubt differ from University to University.

- How these schemes can be incorporated within the existing description for the use of PBE within a crypto-system—see Chapter 9.
- Following from the previous point, how this new description of the crypto-system affects the scenarios governing its use within the Cloud.

## 14.3 Cryptographic Keys

Chapter 8 discussed the different techniques that can be used to construct a Linear Secret Sharing Scheme (LSSS) from boolean formula. These techniques fell into two categories: access tree oriented, or based upon MSPs. It is also known that these different techniques will have some affect upon the composition, namely that of efficiency, of the constructed LSSS. Future investigations should include the comparison of these different techniques looking at how the constructed LSSSs affect: storage requirements, encryption and decryption efficiency.

A similar avenue for further investigation is concerned with key revocation. With Key-Policy schemes it is the Key Authority (KA) that is responsible for the specification and construction of policy rules. These policies can be combined to construct other policy rules, creating a hierarchy of policy rules. Can this be used to facilitate key revocation within Key-Policy schemes?

## 14.4 Use Case Development

The description of how PBE can be used within a crypto-system (given in Chapter 9) does not address fully how other security issues/guarantees such as authenticity and non-repudiation can be assured. This is necessary when discussing the distribution of cryptographic keys and the sending of messages. Area of future work can be the provision of a 'complete description' of a crypto-system that makes used of PBE. Following on from this, another area of investigation will be to look at how such a complete description affects the different scenarios given in Chapter 11. Moreover, these scenarios have been described rather generically and have used 'guiding examples' to aid in their description. An area of future work would be to develop these scenarios further providing more concrete examples of their use.

## 14.5 Practical Considerations

This thesis has primarily been concerned with the theoretical considerations and implications over PBE and the Cloud. Following on from the future work suggested Section 14.5, an obvious next step is to investigate various practical aspects of the use of PBE such as its implementation, deployment and Quality of Service (QoS). Areas of interest that should be addressed include:

- The effect that access policy composition, especially concerning numerical comparisons, has upon the performance of the encryption, decryption and key generation functions.
- The QoS measurements and guarantees that can be made, aside from function efficiency, over PBE.
- The effect that access policy composition has upon, if any, the size of cipher-text produced.
- The representation (encoding) of access policies/rules and list of attributes.
- Existing standards and technologies that should be leveraged or adhered to.
- The construction of a means through which policy rule administration, for both users and CSPs, can be achieved.

# Bibliography

## Access Control

[SC+96] Ravi S. Sandhu, Edward J. Coyne et al. 'Role-Based Access Control Models'. In: *Computer* 29.2 (1996), pp. 38–47. ISSN: 0018-9162. DOI: http://doi.ieeecomputersociety.org/10.1109/2.485845.

[SS94] R S Sandhu and P Samarati. 'Access Control: Principle and Practice'. In: *IEEE COmmunications Magazine* 32.9 (Sept. 1994), pp. 40–48.

[YT05] Eric Yuan and Jin Tong. 'Attributed Based Access Control (ABAC) for Web Services'. In: *Proceedings of the IEEE International Conference on Web Services*. ICWS '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 561–569. ISBN: 0-7695-2409-5. DOI: http://dx.doi.org/10.1109/ICWS.2005.25. URL: http://dx.doi.org/10.1109/ICWS.2005.25.

## Cloud Computing

[AF+09] Michael Armbrust, Armando Fox et al. *Above the Clouds: A Berkeley View of Cloud Computing*. Tech. rep. UCB/EECS-2009-28. Electrical Engineering and Computer Sciences, University of California at Berkeley, Feb. 2009. URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html.

[AHS11] G. Alpár, J.-H. Hoepman and J. Siljee. 'The Identity Crisis. Security, Privacy and Usability Issues in Identity Management'. In: *ArXiv e-prints* (Jan. 2011). arXiv:1101.0427 [cs.CR].

[BCR09] Ken Birman, Gregory Chockler and Robbert van Renesse. 'Toward a cloud computing research agenda'. In: *SIGACT News* 40.2 (2009), pp. 68–80. ISSN: 0163-5700. DOI: http://doi.acm.org/10.1145/1556154.1556172.

[BM03] Philippa J. Broadfoot and Andrew P. Martin. *A Critical Survey of Grid Security Requirements and Technologies*. Tech. rep. PRG-RR-03-15. Wolfson Building Oarks Road Oxford OX1 3QD: Oxford University Computing Laboratory, 2003. URL: http://www.comlab.ox.ac.uk/files/930/RR-03-15.ps.gz.

[CBL08] Monica Chew, Dirk Balfanz and Ben Laurie. '(Under)mining Privacy in Social Networks'. In: *Proceedings of the 2008 IEEE Symposium on Security and Privacy Workshop: Web 2.0 Security and Privacy - W2SP 2008*. Publish Online. 2008. URL: http://w2spconf.com/2008/papers/s3p2.pdf.

[Cri91] Flavin Cristian. 'Understanding fault-tolerant distributed systems'. In: *Commun. ACM* 34.2 (1991), pp. 56–78. ISSN: 0001-0782. DOI: http://doi.acm.org/10.1145/102792.102801.

[DVZ10] Gabriele D'Angelo, Fabio Vitali and Stefano Zacchirolo. 'Content Cloacking: Preserving Privacy with Google Docs and other Web Applications'. To appear in the 25th Symposium On Applied Computing (SAC'10), March 22-26, 2010, Sierre Switzerland. Mar. 2010.

[ENISA-CC-RA09] *Cloud Computing: Benefits, risks and recommendations for information security*. Tech. rep. European Network and Information Security Agency (ENISA), 2009. URL: http://www.enisa.europa.eu/act/rm/files/deliverables/cloud-computing-risk-assessment.

[Fos01] Ian Foster. 'The Anatomy of the Grid: Enabling Scalable Virtual Organizations'. In: *Euro-Par 2001 Parallel Processing* (2001), pp. 1–4. URL: http://dx.doi.org/10.1007/3-540-44681-8_1.

[GTF08] Saikat Guha, Kevin Tang and Paul Francis. 'NOYB: privacy in online social networks'. In: *WOSP '08: Proceedings of the first workshop on Online social networks*. Seattle, WA, USA: ACM, 2008, pp. 49–54. ISBN: 978-1-60558-182-8. DOI: http://doi.acm.org/10.1145/1397735.1397747.

[Hay08] Brian Hayes. 'Cloud computing'. In: *Commun. ACM* 51.7 (2008), pp. 9–11. ISSN: 0001-0782. DOI: http://doi.acm.org/10.1145/1364782.1364786. URL: http://portal.acm.org/citation.cfm?doid=1364782.1364786.

[HM+05] Ragib Hasan, Suvda Myagmar et al. 'Toward a threat model for storage systems'. In: *StorageSS '05: Proceedings of the 2005 ACM workshop on Storage security and survivability.* Fairfax, VA, USA: ACM, 2005, pp. 94–102. ISBN: 1-59593-233-X. DOI: http://doi.acm.org/10.1145/1103780.1103795.

[HS+10] Dan Hubbard, Michael Sutton et al. *Top Threats to Cloud Computing v1.0.* Tech. rep. v1.0. Cloud Security Alliance, Mar. 2010. URL: http://www.cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf.

[JGL08] Meiko Jensen, Nils Gruschka and Norbert Luttenberger. 'The Impact of Flooding Attacks on Network-based Services'. In: *ARES '08: Proceedings of the 2008 Third International Conference on Availability, Reliability and Security.* Washington, DC, USA: IEEE Computer Society, 2008, pp. 509–513. ISBN: 978-0-7695-3102-1. DOI: http://dx.doi.org/10.1109/ARES.2008.16.

[JM09] Carter Jernigan and Behram F.T. Mistree. 'Gaydar: Facebook friendships expose sexual orientation'. In: *First Monday* 14.10 (Oct. 2009). [Online]. URL: http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/2611/2302.

[JS09] Meiko Jensen and Jorg Schwenk. 'The Accountability Problem of Flooding Attacks in Service-Oriented Architectures'. In: *Availability, Reliability and Security, International Conference on* 0 (2009), pp. 25–32. DOI: http://doi.ieeecomputersociety.org/10.1109/ARES.2009.11.

[JS+09] Meiko Jensen, Jorg Schwenk et al. 'On Technical Security Issues in Cloud Computing'. In: *Cloud Computing, IEEE International Conference on* 0 (2009), pp. 109–116. DOI: http://doi.ieeecomputersociety.org/10.1109/CLOUD.2009.60.

[KD+09] Graham Kirby, Alan Dearle et al. *An Approach to Ad hoc Cloud Computing.* Tech. rep. St Andrews Cloud Computing Initiative, School of Computer Science, University of St Andrews, Feb. 2009. URL: http://arxiv.org/abs/1002.4738.

[MA05] Michael McIntosh and Paula Austel. 'XML signature element wrapping attacks and countermeasures'. In: *SWS '05: Proceedings of the 2005 workshop on Secure web services.* Fairfax, VA, USA: ACM, 2005, pp. 20–27. ISBN: 1-59593-234-8. DOI: http://doi.acm.org/10.1145/1103022.1103026.

[MKL09] Tim Mather, Subra Kumaraswamy and Shahed Latif. *Cloud Security and Privacy: An Enterprise Perspective on Risk and Compliance.* Editor Mike Loukides. O'Reilly, 2009.

[MP09] Miranda Mowbray and Siani Pearson. 'A client-based privacy manager for cloud computing'. In: *COMSWARE '09: Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE.* Dublin, Ireland: ACM, 2009, pp. 1–8. ISBN: 978-1-60558-353-2. DOI: http://doi.acm.org/10.1145/1621890.1621897.

[NR05] Syed Naqvi and Michel Riguidel. 'Threat Model for Grid Security Services'. In: *Advances in Grid Computing - EGC 2005.* Ed. by Peter M. A. Sloot, Alfons G. Hoekstra et al. Vol. 3470. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2005, pp. 1048–1055. DOI: 10.1007/11508380_107. URL: http://dx.doi.org/10.1007/11508380_107.

[NS09] Arvind Narayanan and Vitaly Shmatikov. 'De-anonymizing Social Networks'. In: *Security and Privacy, IEEE Symposium on* 0 (2009), pp. 173–187. ISSN: 1081-6011. DOI: http://doi.ieeecomputersociety.org/10.1109/SP.2009.22.

[NW+09] Daniel Nurmi, Rich Wolski et al. 'The Eucalyptus Open-Source Cloud-Computing System'. In: *CCGRID '09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid.* Washington, DC, USA: IEEE Computer Society, 2009, pp. 124–131. ISBN: 978-0-7695-3622-4. DOI: `http://dx.doi.org/10.1109/CCGRID.2009.93`.

[Pea09] Siani Pearson. 'Taking account of privacy when designing cloud computing services'. In: *CLOUD '09: Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing.* Washington, DC, USA: IEEE Computer Society, 2009, pp. 44–52. ISBN: 978-1-4244-3713-9. DOI: `http://dx.doi.org/10.1109/CLOUD.2009.5071532`.

[RT+09] Thomas Ristenpart, Eran Tromer et al. 'Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds'. In: *16th ACM Conference on Computer and Communications Security CCS'09.* Nov. 2009.

[Vou08] Mladen A. Vouk. 'Cloud Computing — Issues, Research and Implementations'. In: *Journal of Computing and Information Technology* 16 (2008), pp. 235–246. DOI: `10.2498/cit.1001391`. URL: `http://cit.srce.unizg.hr/index.php/CIT/article/view/1674/1378`.

[VRM+09] Luis M. Vaquero, Luis Rodero-Merino et al. 'A break in the clouds: towards a cloud definition'. In: *SIGCOMM Comput. Commun. Rev.* 39.1 (2009), pp. 50–55. ISSN: 0146-4833. DOI: `http://doi.acm.org/10.1145/1496091.1496100`.

[WH+10] Gilbert Wondracek, Thorsten Holz et al. 'A Practical Attack to De-anonymize Social Network Users'. In: *Security and Privacy, IEEE Symposium on* 0 (2010), pp. 223–238. ISSN: 1081-6011. DOI: `http://doi.ieeecomputersociety.org/10.1109/SP.2010.21`.

## Legal Perspectives

[Con08] Chris Connolly. 'The US Safe Harbor - Fact or Fiction?' In: *Privacy Laws&Business International* 96 (Dec. 2008). URL: `http://www.galexia.com/public/research/articles/research_articles-pa07.html`.

[Cou09] David A. Couillard. 'Defogging the Cloud: Applying Fourth Amendment Principles to Evolving Privacy Expectations in Cloud Computing'. In: *Minnesota Law Review* 93 (June 2009), pp. 2205–2238.

[JLG08] Paul T. Jaeger, Jimmy Lin and Justin M. Grimes. 'Cloud Computing and Information Policy: Computing in a Policy Cloud?' In: *Journal of Information Technology & Politics* 5.3 (2008), pp. 269–283. URL: `http://www.informaworld.com/10.1080/19331680802425479`.

[Nis04] Helen Nissenbaum. 'Privacy as Contextual Integrity'. In: *Washington Law Review* 79.1 (2004). URL: `http://ssrn.com/abstract=534622`.

[SEC-2002-196] *The application of Commission Decision 520/2000/EC of 26 July 2000 pursuant to Directive 95/46 of the European Parliament and of the Council on the adequate protection of personal data provided by the Safe Harbour Privacy Principles and related Frequently Asked Questions issued by the US Department of Commerce.* English. Commission Staff Working Document SEC (2002) 196. Commission of the European Communities. URL: `http://ec.europa.eu/justice_home/fsj/privacy/docs/adequacy/sec-2002-196/sec-2002-196_en.pdf`.

[SEC-2004-1323] *The implementation of Commission Decision 520/2000/EC on the adequate protection of personal data provided by the Safe Harbour privacy Principles and related Frequently Asked Questions issued by the US Department of Commerce.* English. Commission Staff Working Document SEC (2004) 1323. Commission of the European Communities. URL: `http://ec.europa.eu/justice_home/fsj/privacy/docs/adequacy/sec-2004-1323_en.pdf`.

[Sol07] Daniel J. Solove. ''I've got nothing to hide' and Other Misunderstandings of Privacy'. In: *San Diego Law Review* 44 (2007). GWU Law School Public Law Research Paper No. 289, pp. 745–772. URL: `http://ssrn.com/abstract=998565`.

## Misc Topics on Cryptography

[AL02] Carlisle Adams and Steve Lloyd. *Understanding PKI: Concepts, Standards, and Deployment Considerations.* 2nd. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN: 0672323915.

[Bei96] Amos Beimel. 'Secure Schemes for Secret Sharing and Key Distribution'. PhD thesis. Israel Institute of Technology, 1996.

[BL88] Josh Benaloh and Jerry Leichter. 'Generalized Secret Sharing and Monotone Functions'. In: *Advances in Cryptology —CRYPTO'88* (1988), pp. 27–35. URL: `http://dx.doi.org/10.1007/0-387-34799-2_3`.

[BR08] Mihir Bellare and Phillip Rogaway. 'Code-Based Game-Playing Proofs and the Security of Triple Encryption'. Cryptology ePrint Archive, Report 2004/331. Version: 20081129:034148. 2008. URL: `http://eprint.iacr.org/2004/331`.

[BW07] Dan Boneh and Brent Waters. 'Conjunctive, Subset, and Range Queries on Encrypted Data'. In: *Theory of Cryptography.* Ed. by Salil Vadhan. Vol. 4392. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2007, pp. 535–554. DOI: `10.1007/978-3-540-70936-7_29`. URL: `http://dx.doi.org/10.1007/978-3-540-70936-7_29`.

[Den06] Alexander W Dent. 'Fundamental problems in provable security and cryptography'. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 364.1849 (2006), pp. 3215–3230. DOI: `10.1098/rsta.2006.1895`. eprint: `http://rsta.royalsocietypublishing.org/content/364/1849/3215.full.pdf+html`. URL: `http://rsta.royalsocietypublishing.org/content/364/1849/3215.abstract`.

[ElG85] T. ElGamal. 'A public key cryptosystem and a signature scheme based on discrete logarithms'. In: *IEEE Transactions on Information Theory.* 31 (1985). Could not find PDF, pp. 469–472.

[GMW86] Oded Goldreich, Silvio Micali and Avi Wigderson. 'Proofs that yield nothing but their validity and a methodology of cryptographic protocol design'. In: *Foundations of Computer Science, Annual IEEE Symposium on* 0 (1986), pp. 174–187. ISSN: 0272-5428. DOI: `http://doi.ieeecomputersociety.org/10.1109/SFCS.1986.47`.

[Kan08] Murat Kantarcioglu. 'A Survey of Privacy-Preserving Methods Across Horizontally Partitioned Data'. In: *Privacy-Preserving Data Mining.* Ed. by Ahmed K. Elmagarmid, Amit P. Sheth et al. Vol. 34. Advances in Database Systems. Springer US, 2008, pp. 313–335. ISBN: 978-0-387-70992-5. URL: `http://dx.doi.org/10.1007/978-0-387-70992-5_13`.

[KW93] M Karchmer and A Widgerson. 'On Span Programs'. In: *Proceedings of the Eighth Annual Structure in Complexity Theory Conference, 1993.* IEEE Computer Society, June 1993, pp. 102–111. DOI: `10.1109/SCT.1993.336536`.

[LC10] Zhen Liu and Zhenfu Cao. 'On Efficiently Transferring the Linear Secret-Sharing Scheme Matrix in Ciphertext-Policy Attribute-Based Encryption'. Cryptology ePrint Archive, Report 2010/374. 2010. URL: http://eprint.iacr.org/2010/374.

[RSA78] R. L. Rivest, A. Shamir and L. Adleman. 'A method for obtaining digital signatures and public-key cryptosystems'. In: *Commun. ACM* 21.2 (1978), pp. 120–126. ISSN: 0001-0782. DOI: http://doi.acm.org/10.1145/359340.359342.

[Sha79] Adi Shamir. 'How to share a secret'. In: *Commun. ACM* 22.11 (1979), pp. 612–613. ISSN: 0001-0782. DOI: http://doi.acm.org/10.1145/359168.359176.

[Sti04] Douglas Stinson. 'A Polemic on Notions of Cryptographic Security'. July 2004. URL: http://www.cacr.math.uwaterloo.ca/~dstinson/papers/polemic.pdf.

## Guides to Pairing Based Cryptography

[Bon07] Dan Boneh. 'A Brief Look at Pairings Based Cryptography'. In: *Foundations of Computer Science, Annual IEEE Symposium on* 0 (2007), pp. 19–26. ISSN: 0272-5428. DOI: http://doi.ieeecomputersociety.org/10.1109/FOCS.2007.51.

[DBS04] Ratna Dutta, Rana Barua and Palash Sarkar. 'Pairing-Based Cryptographic Protocols : A Survey'. Cryptology ePrint Archive. Version 20040624:121914. 2004. URL: http://eprint.iacr.org/2004/064.

[Men09] Alfred Menezes. 'Recent Trends in Cryptography'. In: ed. by Ignacio Luengo. Vol. 477. American Mathematical Society and Real Sociedad Matemática Española, 2009. Chap. An Introduction to Pairing-Based Cryptography, pp. 47–65.

## Predicate Based Cryptography

[AL+10] Joseph A. Akinyele, Christoph U. Lehmann et al. 'Self-Protecting Electronic Medical Records Using Attribute-Based Encryption'. Cryptology ePrint Archive, Report 2010/565. Version 20101118:220821. 2010. URL: http://eprint.iacr.org/2010/565.

[BBG05] Dan Boneh, Xavier Boyrn and Eu-Jin Goh. 'Hierarchical identity based encryption with constant size ciphertext'. In: *Lecture Notes in Computer Science* 3494 (2005). Anglais, p. 17.

[BDC+04] Dan Boneh, Giovanni Di Crescenzo et al. 'Public Key Encryption with Keyword Search'. In: *Advances in Cryptology - EUROCRYPT 2004* 3027/2004 (2004), pp. 506–522. URL: http://www.springerlink.com/content/0hafhrbbvt2l7vn3.

[BF01] Dan Boneh and Matt Franklin. 'Identity-Based Encryption from the Weil Pairing'. In: *Advances in Cryptology — CRYPTO 2001* 2139/2001 (2001), pp. 213–229. DOI: 10.1007/3-540-44647-8_13.

[BF+10] Rakesh Bobba, Omid Fatemieh et al. 'Attribute-Based Messaging: Access Control and Confidentiality'. In: *ACM Trans. Inf. Syst. Secur.* 13 (4 Dec. 2010), 31:1–31:35. ISSN: 1094-9224. DOI: http://doi.acm.org/10.1145/1880022.1880025. URL: http://doi.acm.org/10.1145/1880022.1880025.

[BKP10] Rakesh Bobba, Himanshu Khurana and Manoj Prabhakaran. 'Attribute-Sets: A Practically Motivated Enhancement to Attribute-Based Encryption'. In: *Computer Security — ESORICS 2009* (2010), pp. 587–604. URL: http://dx.doi.org/10.1007/978-3-642-04444-1_36.

[BSW07] John Bethencourt, Amit Sahai and Brent Waters. 'Ciphertext-Policy Attribute-Based Encryption'. In: *SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 321–334. ISBN: 0-7695-2848-1. DOI: http://dx.doi.org/10.1109/SP.2007.11.

[BSW10] Dan Boneh, Amit Sahai and Brent Waters. 'Functional Encryption: Definitions and Challenges'. Cryptology ePrint Archive, Report 2010/543. Version 20101025:151309. 2010. URL: http://eprint.iacr.org/2010/543.

[Cha07] Melissa Chase. 'Multi-authority Attribute Based Encryption'. In: *Theory of Cryptography* (2007), pp. 515–534. URL: http://dx.doi.org/10.1007/978-3-540-70936-7_28.

[Coc01] Clifford Cocks. 'An Identity Based Encryption Scheme Based on Quadratic Residues'. In: *Cryptography and Coding* (2001), pp. 360–363. DOI: 10.1007/3-540-45325-3_32.

[EM+09] Keita Emura, Atsuko Miyaji et al. 'A Ciphertext-Policy Attribute-Based Encryption Scheme with Constant Ciphertext Length'. In: *Information Security Practice and Experience* (2009), pp. 13–23. URL: http://dx.doi.org/10.1007/978-3-642-00843-6_2.

[Gen06] Craig Gentry. 'Practical Identity-Based Encryption Without Random Oracles'. In: *Advances in Cryptology - EUROCRYPT 2006* (2006), pp. 445–464. URL: http://dx.doi.org/10.1007/11761679_27.

[GS+06] Vipul Goyal, Amit Sahai et al. 'Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data'. In: *Conference on Computer and Communications Security: Proceedings of the 13th ACM conference on Computer and communications security.* Oct. 2006.

[IP+09] Luan Ibraimi, Milan Petkovic et al. *Ciphertext-Policy Attribute-Based Threshold Decryption with Flexible Delegation and Revocation of User Attributes (extended version).* Tech. rep. TR-CTIT-09-12. Enschede: Centre for Telematics and Information Technology, University of Twente, Apr. 2009. URL: http://doc.utwente.nl/65471/.

[KSW08] Jonathan Katz, Amit Sahai and Brent Waters. 'Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products'. In: *Advances in Cryptology – EUROCRYPT 2008* (2008), pp. 146–162. URL: http://dx.doi.org/10.1007/978-3-540-78967-3_9.

[LC+09] Xiaohui Liang, Zhenfu Cao et al. 'Attribute based proxy re-encryption with delegating capabilities'. In: *ASIACCS '09: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security.* Sydney, Australia: ACM, 2009, pp. 276–286. ISBN: 978-1-60558-394-5. DOI: http://doi.acm.org/10.1145/1533057.1533094.

[LC+10] Huang Lin, Zhenfu Cao et al. 'Secure threshold multi authority attribute based encryption without a central authority'. In: *Information Sciences* 180.13 (2010), pp. 2618–2632. ISSN: 0020-0255. DOI: DOI:10.1016/j.ins.2010.03.004. URL: http://www.sciencedirect.com/science/article/B6V0C-4YK7J6G-4/2/67406d4044d9006f6c0f2c6520e3cda9.

[LO+10] Allison Lewko, Tatsuaki Okamoto et al. 'Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption'. Cryptology ePrint Archive, Report 2010/110. http://eprint.iacr.org/. 2010.

[LW10] Allison Lewko and Brent Waters. 'Decentralizing Attribute-Based Encryption'. Cryptology ePrint Archive, Report 2010/351. 2010. URL: http://eprint.iacr.org/2010/351.

[MKE09] Sascha Muller, Stefan Katzenbeisser and Claudia Eckert. 'On multi-authority ciphertext-policy attribute-based encryption'. In: *Bulletin of the Korean Mathematical Society* 46.4 (2009), pp. 803–819.

[MPR10] Hementa K. Maji, Manoj Prabhakaran and Mike Rsulek. 'Attribute-Based Signatures'. Cryptology ePrint Archive, Report 2010/595. Version 20101124:045114. Nov. 2010. URL: `http://eprint.iacr.org/2010/595`.

[OSW07] Rafail Ostrovsky, Amit Sahai and Brent Waters. 'Attribute-based encryption with non-monotonic access structures'. In: *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security.* Alexandria, Virginia, USA: ACM, 2007, pp. 195–203. ISBN: 978-1-59593-703-2. DOI: `http://doi.acm.org/10.1145/1315245.1315270`.

[PT+06] Matthew Pirretti, Patrick Traynor et al. 'Secure attribute-based systems'. In: *Conference on Computer and Communications Security: Proceedings of the 13th ACM conference on Computer and communications security.* 2006.

[Sha85] Adi Shamir. 'Identity-Based Cryptosystems and Signature Schemes'. In: *Advances in Cryptology* (1985), pp. 47–53. DOI: `10.1007/3-540-39568-7_5`.

[SSW09] Emily Shen, Elaine Shi and Brent Waters. 'Predicate Privacy in Encryption Systems'. In: *TCC '09: Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography.* San Francisco, CA: Springer-Verlag, 2009, pp. 457–473. ISBN: 978-3-642-00456-8. DOI: `http://dx.doi.org/10.1007/978-3-642-00457-5_27`.

[SW05] Amit Sahai and Brent Waters. 'Fuzzy Identity-Based Encryption'. In: *Advances in Cryptology – EUROCRYPT 2005* (2005), pp. 457–473. URL: `http://dx.doi.org/10.1007/11426639_27`.

[Wat10] Brent Waters. 'Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization'. Cryptology ePrint Archive, Report 2008/290. Version 20101220:203013. 2010. URL: `http://eprint.iacr.org/2008/290`.

[YF+04] Danfeng Yao, Nelly Fazio et al. 'ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption'. In: *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security.* Washington DC, USA: ACM, 2004, pp. 354–363. ISBN: 1-58113-961-6. DOI: `http://doi.acm.org/10.1145/1030083.1030130`.

## Online

[ACLU-fb-app] *What Do Quizzes Really Know About You?* English. Facebook Application Page. Northern California Chapter of the American Civil Liberties Union. URL: `http://www.facebook.com/apps/application.php?id=114232425072`.

[Arr09] Michael Arrington. *Celebrity Twitter Accounts Hacked (Bill O'Reilly, Britney Spears, Obama, More).* English. TechCrunch. Jan. 2009. URL: `http://techcrunch.com/2009/01/05/either-fox-news-had-their-twitter-account-hacked-or-bill-oreilly-is-gay-or-both/`.

[AWS] English. Amazon Web Services LLC. URL: `http://aws.amazon.com/`.

[AWS-Pricing] English. Amazon Web Services LLC. URL: `http://aws.amazon.com/ec2/pricing/`.

[Bar10] Emma Barnett. *Facebook's Mark Zuckerberg says privacy is no longer a 'social norm'.* English. Telegraph Media Group Limited. Jan. 2010. URL: `http://www.telegraph.co.uk/technology/facebook/6966628/Facebooks-Mark-Zuckerberg-says-privacy-is-no-longer-a-social-norm.html`.

[CardSpace] English. Microsoft Inc. URL: `http://www.microsoft.com/windows/products/winfamily/cardspace/`.

[For10] Chris Foresman. *Brief Facebook glitch sent private messages to wrong users*. English. Ars Technica. Feb. 2010. URL: http://arstechnica.com/tech-policy/news/2010/02/brief-facebook-glitch-sent-private-messages-to-wrong-users.ars.

[GooApps] English. Google Inc. URL: http://www.google.com/apps/intl/en/business/index.html.

[Hed10] Marc Hedlund. *Protecting "Cloud" Secrets with Grendel*. English. Wesabe, Inc. Jan. 2010. URL: http://blog.wesabe.com/2010/01/04/protecting-cloud-secrets-with-grendel/.

[HigginsProject] English. The Eclipse Foundation. URL: http://www.eclipse.org/higgins/.

[Ho10] Colin Ho. *Apache flaw opens systems up to attack*. English. ZDNet UK. Mar. 2010. URL: http://www.zdnet.co.uk/news/security-threats/2010/03/08/apache-flaw-opens-systems-up-to-attack-40077943/.

[Lem08] Robert Lemos. *Alliance forms to fix DNS poisoning flaw*. English. Security Focus. July 2008. URL: http://www.securityfocus.com/news/11526.

[OpenID] English. OpenID Foundation. URL: http://openid.net/.

[Per09] Sarah Perez. *How Safe are Facebook Applications?* English. Read Write Web. Oct. 2009. URL: http://www.readwriteweb.com/archives/how_safe_are_facebook_applications.php.

[SafeHarbour] English. U.S. Department Of Commerce. URL: http://www.export.gov/safeharbor/.

[Tho09] Roger Thompson. *Hacked Facebook Applications reach out to Exploit Sites in Russia*. English. AVG. Oct. 2009. URL: http://thompson.blog.avg.com/2009/10/hacked-facebook-applications-reach-out-to-exploit-sites-in-russia.html.

[Vas09] Jessica E. Vascellaro. *Google Discloses Privacy Glitch*. English. Wall Street Journal. Mar. 2009. URL: http://blogs.wsj.com/digits/2009/03/08/1214/.

[Won10] Phil Wong. *Conversations About the Internet #5: Anonymous Facebook Employee*. English. The Rumpus. Jan. 2010. URL: http://therumpus.net/2010/01/conversations-about-the-internet-5-anonymous-facebook-employee.

[XSRF10] *The Cross-Site Request Forgery (CSRF/XSRF) FAQ*. English. CGI Security. Apr. 2010. URL: http://www.cgisecurity.com/csrf-faq.html.

[XSS02] *The Cross-Site Scripting (XSS) FAQ*. English. CGI Security. May 2002. URL: http://www.cgisecurity.com/xss-faq.html.

# Content Listings and Glossary

# List of Acronyms

**AA** Attribute Authority.

**ABAC** Attribute Based Access Control.

**ABE** Attribute Based Encryption.

**ABS** Attribute Based Signing.

**AH** Attribute Hiding.

**API** Application Programming Interface.

**BSW** Bethencourt Sahai Waters.

**CoClo** Content Cloaking.

**CP** Ciphertext-Policy.

**CSP** Cloud Service Provider.

**DABE** Distributed Attribute Based Encryption.

**DDI** Duty Delegation Infrastructure.

**DecAuth** Decryption Authority.

**DKG** Decryption Key Generator.

**EA** Escrow Authority.

**EncAuth** Encryption Authority.

**FBE** Functional Based Encryption.

**IaaS** Infrastructure as a Service.

**IBE** Identity Based Encryption.

**KA** Key Authority.

**KP** Key-Policy.

**LBAC** Lattice Based Access Control.

**LSSS** Linear Secret Sharing Scheme.

**MCP** Multicast Content Provision.

**MKG** Master Key Generator.

**MSP** Monotone Span Program.

**NOYB** None of Your Business.

**OSN** Online Social Network.

**PA** Policy Authority.

**PaaS** Platform as a Service.

**PBE** Predicate Based Encryption.

**PDP** Policy Decision Point.

**PEP** Policy Enforcement Point.

**PH** Payload Hiding.

**PKI** Public Key Infrastructure.

**QoS** Quality of Service.

**RA** Revocation Authority.

**RBAC** Role Based Access Control.

**SaaS** Software as a Service.

**SSS** Secret Sharing Scheme.

# List of Figures

# List of Definitions