PHD

Identity as a Service: An adaptive security infrastructure and privacy-preserving user identity for the cloud environment

Vo, Hoang Tri

Award date:
2020

Awarding institution:
University of Plymouth

UNIVERSITY OF
PLYMOUTH

IDENTITY AS A SERVICE

AN ADAPTIVE SECURITY INFRASTRUCTURE AND PRIVACY-PRESERVING USER IDENTITY FOR THE CLOUD ENVIRONMENT

by

HOANG TRI VO

A thesis submitted to the University of Plymouth
in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

School of Engineering, Computing, and Mathematics

November 2019

ACKNOWLEDGEMENTS

The doctoral research was a journey beginning with a blurred picture, uncertain questions, and multiple paths to take. I can imagine as if I were at a train station. The start station was empty. I was lucky to meet Prof Dr Woldemar Fuhrmann, my Director of Studies. He gave me a ticket to the train and showed me the right path to go. These baby steps were important that helped me to walk on my own. During my trip, I met Dr Fischer-Hellmann, my first supervisor, who commented on my thesis and helped me to improve my writing skills. While answering his questions, the final picture becomes clearer to me. I wish to thank Prof Dr Steven Furnell, my second supervisor, who met me at the end of each station. He took over the proofreading within an unbelievably short time and provided invaluable comments for restructuring the thesis. Words can neither qualify nor quantify your guidance and useful advice.

I also wish to thank my boss at Deutsche Telekom, Bodo Endrehs, for his mental support.

I was not alone during my trip and have companions as well. I wish to thank my parents, my younger brother, and especially my seven-year-old son (for being around). I had precious moments with him in the last four years. In general, the research was joyful, and I continued until I arrived at the final station of the journey.

Sincerely thank you.

AUTHOR'S DECLARATION

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award without prior agreement of the Doctoral College Quality Sub-Committee.

Work submitted for this research degree at the University of Plymouth has not formed part of any other degree either at the University of Plymouth or at another establishment.

This study was sponsored by the part-time study programs Bologna at Deutsche Telekom. The company gave the author 10 holidays a year for doing research.

Publications:

1. Identity-as-a-Service (IDaaS): A missing gap for moving enterprise applications in Inter-Cloud.
2. How to adapt authentication and authorization infrastructure of applications for the cloud.
3. Privacy-preserving user identity in Identity-as-a-Service.
4. Automated trust negotiation for cloud applications in Identity-as-a-Service.
5. Efficient privacy-preserving user identity with Purpose-based Encryption.
6. Identity-as-a-Service: An adaptive security infrastructure and privacy-preserving user identity for the cloud environment.

Conferences and journal:

1. The 11th International Networking Conference (INC 2016).
2. The 5th IEEE International Conference on Future Internet of Things and Cloud (FiCloud 2017).
3. The 21st IEEE Conference on Innovation in Clouds, Internet, and Networks (ICIN 2018).
4. The 2019 IEEE International Conference on Advanced Communication Technologies and Networking (CommNet) – Security.
5. The 2019 IEEE International Symposium on Networks, Computers and Communications (ISNCC'19): Trust, Security, and Privacy.
6. The Future Internet Journal: Security and Privacy in Information and Communication Systems.

Word count of main body of thesis: 37429.

Signed .....

Date .....

Abstract

Identity-as-a-Service: An Adaptive Security Infrastructure and Privacy-Preserving User Identity for the Cloud Environment

Hoang Tri Vo

In recent years, enterprise applications have begun to migrate from a local hosting to a cloud provider and may have established a business-to-business relationship with each other manually. Adaptation of existing applications requires substantial implementation changes in individual architectural components. Existing work has focused on migrating the applications with functional and non-functional aspects. However, none of them has focused so far on the adaptation of the required security infrastructure. On the other hand, users may store their Personal Identifiable Information (PII) in the cloud environment so that cloud services may access and use it on demand. Although cloud services specify their privacy policies, it is not possible to guarantee that they will follow their policies and will not (accidentally) transfer PII to another party. To solve the aforementioned issues, this thesis presents Identity-as-a-Service (IDaaS) as a trusted Identity and Access Management with two new requirements:

Firstly, IDaaS adapts the required security infrastructures of cloud services to complete a business-to-business transaction on demand. The thesis decouples the security infrastructures from the business logic of the applications and models them as a security topology. When the business comes up with a new e-commerce scenario, IDaaS uses the security topology to adapt the security infrastructures of cloud services and propagate PII from the original caller via intermediaries to the end service on demand. Also, when cloud services migrate to other cloud providers, the trust relationship between them is adapted and preserved during the migration. As a result, developers do not need to re-implement their applications upon each change. The security infrastructure is portable across cloud providers as well as interoperable with the protected cloud services in the backend.

Secondly, the thesis proposes a novel Purpose-based Encryption to protect the confidentiality of PII in federated security domains. Unlike prior research, the thesis involves the least user interaction to prevent identity theft via the human link. By using Purpose-based Encryption, users encrypt PII with their intended purposes and in a given period and disseminate the ciphertext in federated security domains. To complete a business transaction, cloud services can decrypt the ciphertext for the right purposes and in the given time, but nothing more. As a result, the encryption protects the disclosure of PII over intermediaries in a business transaction and against untrusted hosts. The solution is compliant with the General Data Protection Regulation of the European Union. The implementation can be easily adapted to existing Identity Management systems, and the performance is fast.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF LISTINGS

LIST OF ABBREVIATIONS AND ACRONYMS

AAI	Authentication and Authorisation Infrastructure
ABAC	Attribute-based Access Control
ABE	Attribute-based Encryption
AES	Advanced Encryption Standard
B2B	Business-to-business
CED	Computing with Encrypted Data
CEDEF	Computing with Encrypted Data and Encrypted Function
CP-ABE	Ciphertext-Policy Attribute-based Encryption
DLP	Discrete Logarithm Problem
DoB	Date of Birth
EC	Elliptic Curve
EU	European Union
FIDM	Federated Identity Management
FE	Functional Encryption
FHE	Fully Homomorphic Encryption
GDPR	General Data Protection Regulation
IBE	Identity-based Encryption
IDM	Identity Management
IdP	Identity Provider
ISV	Independent Software Vendor
JAAS	Java Authentication and Authorization Service
JASPIC	Java Authentication Service Provider Interface for Containers
JEE	Java Enterprise Edition
KP-ABE	Key-Policy Attribute-based Encryption
LSSS	Linear Secret Sharing Scheme

NIST	National Institute for Standards and Technology
OAuth	Open Authorization
P3P	Platform for Privacy Preferences Project
PBE	Purpose-based Encryption
PAT	Purpose Access Token
PBAC	Purpose-based Access Control
PDP	Policy Decision Point
PE	Predicate Encryption
PEP	Policy Enforcement Point
PII	Personal Identifiable Information
RBAC	Role-based Access Control
RSA	Rivest-Shamir-Adleman
SaaS	Software-as-a-Service
SAML	Security Assertion Markup Language
SC	Service Consumer
SCIM	System for Cross-domain Identity Management
SOA	Service-oriented Architecture
SOAP	Simple Object Access Protocol
SP	Service Provider
SSO	Single-Sign-On
STS	Security Token Service
TAT	Time Access Token
TOSCA	Topology and Orchestration Specification for Cloud Applications
TTP	Trusted Third Party
XACML	eXtensible Access Control Markup Language
ZKP	Zero Knowledge Proof

Chapter 1

Introduction and overview

1.1 Introduction

In a local hosting environment, traditional applications have their own implementation for authentication and authorization. One local application has *n* user accounts. The management process of user identities associated with different levels of access control gave birth to Identity Management (IDM).

In Federated Identity Management, service providers (SPs) may integrate with each other based on shared IDM. Shared IDM allows providers from different security domains to exchange messages containing authentication and authorisation credentials about users [1]. As a result, users maintain one-login identity and access resources from multiple SPs. In additions, it reduces the cost for SPs because they no longer manage users that are not under their control (i.e., users may be under the government of a partner service) [2].

In cloud computing, enterprise applications come from federated security domains; provide themselves as an SP and cooperate with each other. From the beginning, they might adapt their local IDM with one another manually. However as time goes by, their applications or their partner service migrate to another cloud provider, so they need to re-implement the security mechanisms to grant new access controls upon each change. In such a dynamic provisioning environment on the cloud, enterprise applications may prefer outsourcing their security implementation to a third party central service [3]. In this case, their security can be strengthened by a specialised provider and reduce their cost.

A traditional authorisation system is Role-based Access Control (RBAC) [4], whereby access control depends on the role of an entity after it authenticates to the system. Developers of the application also tend to hardcoding the authorisation logics into the source code of the program and only these

developers can understand it. Authorisation hardcoding is in contrast to a policy-based approach, in which administrators can manage different rules and dynamically changed interpreted logic without modifying the underlying implementation [5]. Without the policy-based approach, it is a big obstacle for applications to be portable on the clouds, because developers have to adapt source code to various security domains with different security policies.

The Attribute-based Access Control (ABAC) is a policy-based approach for fine-grained authorisation because access control in this approach does not require identifying the entity as a whole, but depends on its attributes [6]. Gartner predicts: “By 2020, 70% of all businesses will use ABAC as the dominant mechanism to protect critical assets” [7]. This is the reason why we focus on IDM not only for the traditional RBAC but also for ABAC. Existing work adapted functional components of applications to the cloud environment [8]. The adaptation for authentication and authorisation has security challenges and needs more research.

From the user’s perspective, users may have multiple identities with public and private profiles on the Internet. They may prefer to access multiple SPs in federated security domains, but also preserve their privacy. When users do not store their personal identities in a local machine, but in an SP in the cloud, they may be interested in questions such as where their data are stored actually, how secure it is, who can access it except themselves. Even if SPs specified their privacy policies, it is not possible to guarantee that they will follow their policies and will not (accidentally) transfer user data to another party. If users cannot trust any third parties to hold their data, they might prefer to trust themselves. In a user-centric approach, users actively decide which identity credentials they want to exchange with which SPs. They can selectively

disclose a minimal attribute set [9] or hide their identities by using *anonymous credentials* in transactions with SPs [10] [11]. However, the user-centric approach has a limitation for the users themselves because it overloads the IDM tasks to the user's decision in every transaction. Moreover, anonymous credentials require SPs to accept it and to develop an adaptation. However, most SPs, who select an Identity Provider (IdP) based on how much information they can collect from the user [12], are not likely to support anonymous credentials.

The open standards organisation, OASIS, mentioned Identity-as-a-Service (IDaaS) as an approach to Identity Management in which an entity (individual or organization) relies on special service provider's functionalities that allows the entity to perform an electronic transaction, which requires identity data managed by this provider [13]. Since the definition is non-standard and coarse, the following thesis revisits the requirements of a traditional IDM system by Kim Cameron [14] and specifies which requirements are still missing in cloud computing. An IDM system is much likely to succeed when it benefits all parties, including users, SPs and IdP [12]. This is the goal of the thesis to propose such a model for IDaaS.

The aim of this research is to propose a novel reference architecture of IDaaS for the cloud environment. The thesis defines two novel requirements for IDaaS bringing benefits to both SPs and users as follows:

IDaaS is a trusted Identity and Access Management that (1) adapts the authentication and authorisation infrastructures of cloud services to complete a business-to-business (B2B) transaction on demand and (2) protects the dissemination of user identity in federated security domains.

In order to achieve this aim, several issues need to be thoroughly investigated and analysed, and the research addresses this via the following objectives:

1. To collect existing design patterns of IDM in order to understand the missing gap in the current design architectures in general. From this missing gap, it could propose a suitable reference architecture for IDaaS.
2. To analyse the adaptation obstacles and existing effort when an application component migrates from local hosting to a cloud provider and between cloud providers, especially the adaptation of the required security mechanisms. From the current states, it may propose a holistic approach to adapt the authentication and authorisation infrastructure of application components for the cloud environment.
3. To adapt the authentication and authorisation infrastructures of an application component, it is necessary to understand what developers normally implement to protect their applications. Therefore, the thesis collects the most frequently used security design patterns that protect an application. This collection of design patterns may help developers to save their working effort when they implement security mechanisms repeatedly in various applications for heterogeneous environments.
4. To revise the effort of privacy-preserving user identity that targets a large distributed and heterogeneous environment for the cloud environment. It means a solution that works in one security domain is not enough. It is worth mentioning that human-PC link is the weakest link compared to the links between the PC, SP, and IdP [1]. Therefore, the human link is the major target of identity theft [1]. For this reason, the literature review also focuses on solutions that reduce human interaction from identity disclosure as much as possible. Furthermore, absolute protection does not exist. Thus, the

research seeks an efficient solution that is compliant with the law, in particular, the General Data Protection Regulation (GDPR) [15].

5. To evaluate the proposed solution in a testbed with multiple cloud providers for the adaptation of the required security infrastructures. It also evaluates, to which extent the user privacy is preserved during the lifecycle of the user identity in the testbed. The research also evaluates the proposed solution, if it can be easily integrated into existing IDM systems on the market, and if the performance is acceptable for completing a usual business transaction.

1.2 Thesis organisation

The thesis is organised as in Figure 1. The structure reflects the progression and development of the research, with the initial background review of the state-of-the-art enabling the establishment of technology foundations upon which to build the new contributions. This then leads into the main contributions of the research, with a novel architectural approach that is then implemented and evaluated in practice.

Chapter 2 summaries the fundamentals of IDM and then describes the motivation scenarios for Identity-as-a-Service. From these scenarios, Section 2.3 identifies the challenges that the research will encounter.

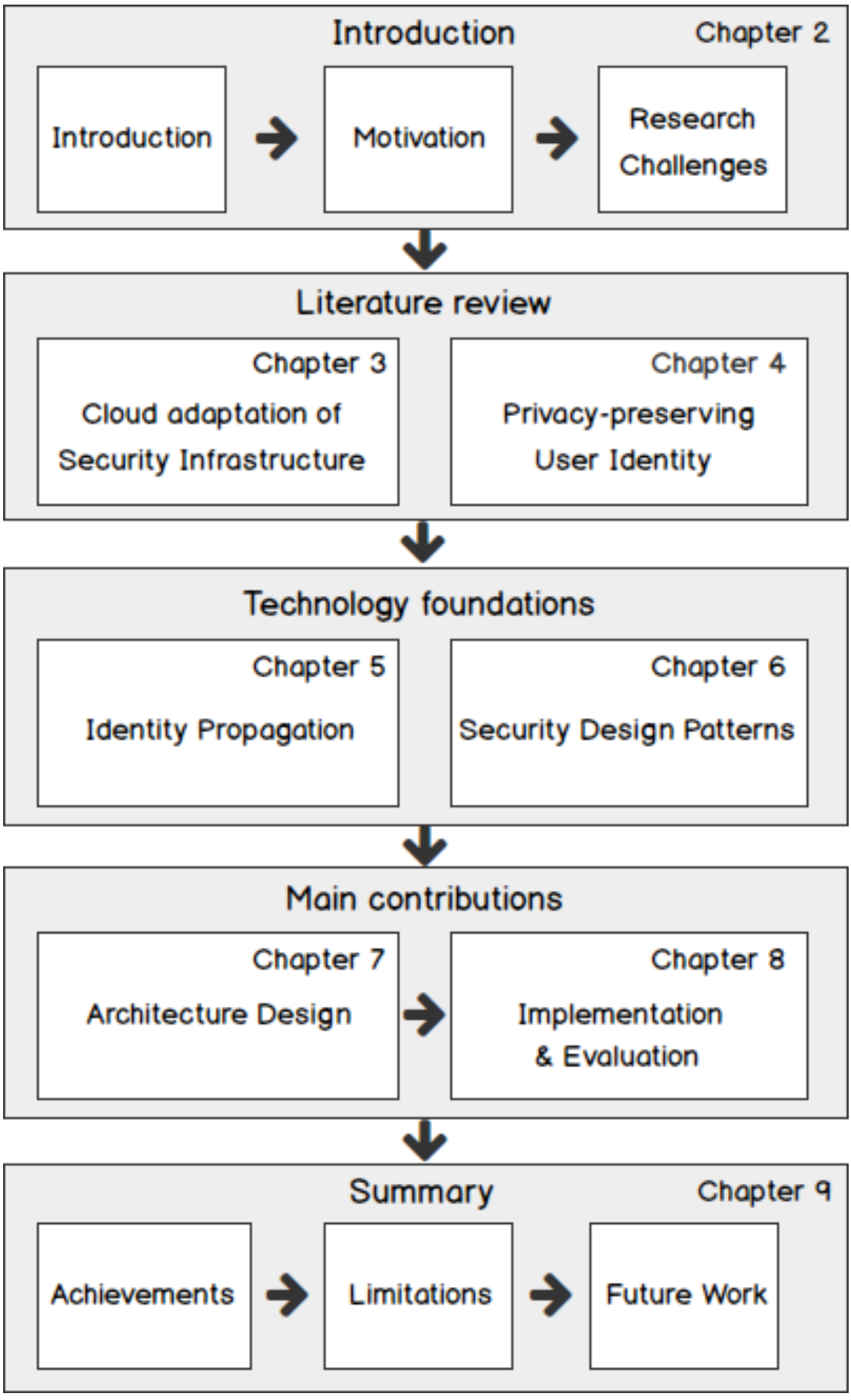

Figure 1: Thesis organisation

One of the research challenges is to adapt the required security infrastructure of an application for the cloud environment. 3 focuses on reviewing the literature for this challenge. Section 3.1 first reviews existing design patterns of IDM and analyses the missing gap in them. Then Section 3.2.1 and Section 3.2.2 review the literature for the adaptation effort with functional and security aspects, respectively.

Another research challenge is to protect user privacy for the cloud environment. Chapter 4 focuses on reviewing the literature on privacy-preserving user identity. Section 4.1 first explains the properties of user privacy. Afterwards, Section 4.2 reviews related work that preserves these properties. In particular, it revisits the user-centric approach that preserves user identity over an intended channel and the confidentiality approach that protects user identity over an unintended channel. Then Section 4.3 reviews related work of *Purpose-based Access Control* and explain why researchers choose it to control the disclosure of PII. Moreover, Section 4.4 reviews *secure computation* such as *Garbled Circuits* and *Homomorphic Encryption* that compute a function on encrypted data on an untrusted host. This section explains why researchers cannot use these solutions. At the end of the chapter, Section 4.5 gives a literature review on *Functional Encryption* (FE). This section explains how researchers choose a subclass of FE, *Attribute-based Encryption*, which satisfies the needs to protect PII on an untrusted host.

To save the working effort of developers to adapt their applications implementations for a business scenario, it is necessary to understand the identity propagation between SPs in a call chain. Chapter 5 reviews the design patterns for identity propagation. These patterns clarify how PII is transmitted between entities to complete a business transaction. In particular, it explains

how a user identity is *impersonated* (Section 5.2), *forwarded* (Section 5.3), and *delegated* (Section 5.4) between entities.

To save the working effort of developers to adapt the implementations to protect their application components, it is necessary to understand how developers usually implement an authentication and authorisation for their applications. Chapter 6 reviews the design patterns for authentication (Section 6.1) and authorisation (Section 6.2). In addition, it reviews how to decouple the authentication and authorisation implementations from the application logic using an *intercepting Web agent* (Section 6.3.1) and a *security gateway* (Section 6.3.2), and how to implement them in scripting languages, in Java EE, and in .NET Frameworks.

Chapter 7 presents the architecture designs of the thesis. Section 7.1 first gives an overview of the thesis and proposes a novel reference architecture for IDaaS. Then Section 7.2 and Section 7.3 dive deeper into the concept of privacy-preserving user identity and the concept of the security infrastructures adaptation, respectively. In each section, researchers explain the architecture considerations for their decisions in Section 7.2.1 and 7.3.1.

Chapter 8 describes the implementation and evaluation of the solution concept. The first part of Chapter 8 evaluates the solution for privacy-preserving user identity in the testbed. In particular, Section 8.1.1 and Section 8.1.2 describe the testbed and evaluate a lifecycle of PII in two federated domains, respectively. Then Section 8.1.3 and Section 8.1.4 describe the implementation as well as evaluate the performance and discusses the limitations of IDaaS. The second part of Chapter 8 focuses on the adaptation of security infrastructures. In particular, Section 8.2.1 demonstrates the life cycle of the security infrastructure

from the development phase to the migration phase. It shows two cloud services can establish trust with each other when they migrate from a local hosting to a cloud provider and between multiple cloud providers (e.g., OpenStack and AmazonWS). Afterwards, Section 8.2.3 and Section 8.2.4 describe the implementation of the required security components. Finally, Section 8.2.5 evaluates the portability and interoperability issue of the adaptation effort.

Chapter 9 summarises the achievements as well as the limitations of the thesis and discuss future work.

Chapter 2

Identity-as-a-Service overview and issues

This chapter first summarises the fundamentals of IDM. Section 2.2 then describes the motivation scenarios for Identity-as-a-Service. From these scenarios, Section 2.3 identifies the challenges that the research will encounter.

2.1 An overview of Identity Management

In a local hosting environment, traditional applications have their own implementations for authentication and authorisation. Each application has *n* user accounts. The management process of *n* user accounts from multiple applications associated with different levels of access control gave birth to Identity Management (IDM). The following sections introduce an overview of IDM.

2.1.1 Identity Management fundamentals

2.1.1.1 Personal Identifiable Information (PII)

Personal Identifiable Information (PII) or *user identity* is information about a person (e.g., name, age, addresses), which makes it possible to identify him or her [16]. IDM is the process of managing PII with some basic functionalities such as administration, authentication, policy enforcement, and assertions [16].

An *identifier* is an attribute that uniquely identifies a user in a given domain so that no other users have the same identifier in this domain. It means a user may have a different identifier in another domain. For privacy concern, users may not disclose their identifiers to another domain, because they may worry that this information can uniquely identify them, while a common user attribute (e.g., hair colour) typically cannot [1].

2.1.1.2 Authorisation credentials

An *attribute assertion* is a claim asserted by an authoritative source to say that a user possesses a certain PII. An *Identity Provider* (IdP) is an authoritative source that authenticates a user and issues a digitally signed attribute assertion about the authenticated user (i.e., an *authorisation credential*) [1]. Figure 2 illustrates the relationship between an IdP and Service Providers (SPs) in one domain. After an IdP authenticates a user, it issues an authorisation credential and gives it to an SP. SPs are *relying parties* of the given IdP because they trust the IdP to issue valid credentials, for which the IdP is authoritative [1]. SPs may require PII to authorise a user request, to complete a business transaction, or to customise their services [17].

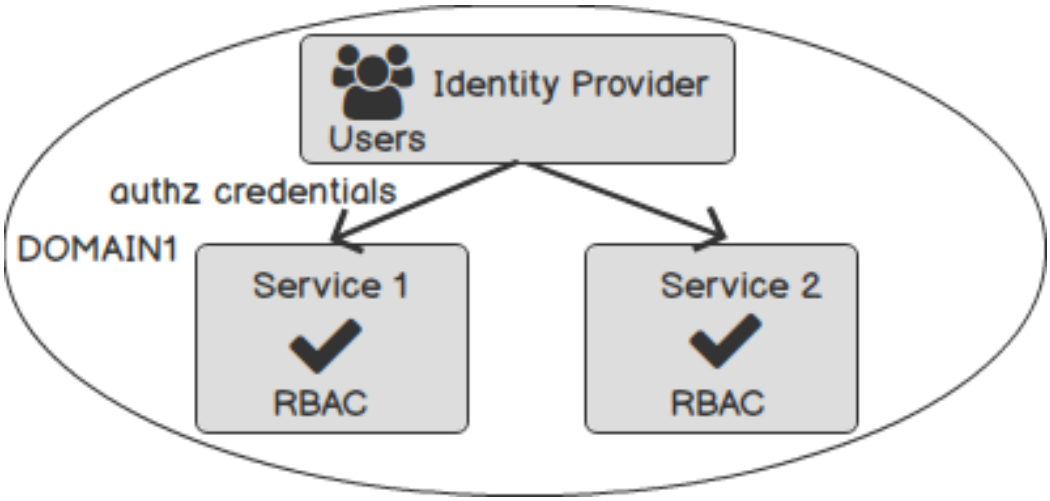

Figure 2: Identity Provider and relying parties in one domain.

2.1.1.3 Authorisation

After the SP receives an authorisation credential, it decides if the authenticated user has access to its resource or not. In a traditional authorisation system, common access control is *Role-based Access Control* (RBAC) [4], whereby access control depends on the roles of the user after he or she authenticates to the system. The *Attribute-based Access Control* (ABAC) is a policy-based approach for fine-grained authorisation. In ABAC, access control does not

require identifying the user but depends on user attributes. For example, a user can buy a specific DVD if he or she is older than 18 years old. In this example, access control is based on the user's age. The *eXtensible Access Control Markup Language* (XACML) [6] is a standard policy language that supports ABAC. Figure 3 shows a reference architecture of XACML that includes four main components: a *Policy Decision Point* (PDP) evaluates an access request against the policies, a *Policy Enforcement Points* (PEP) enforces authorisation decision from the PDP, a *Policy Information Point* (PIP) stores and collects missing user attributes from external resources, and a *Policy Administration Point* (PAP) administrates the policies.

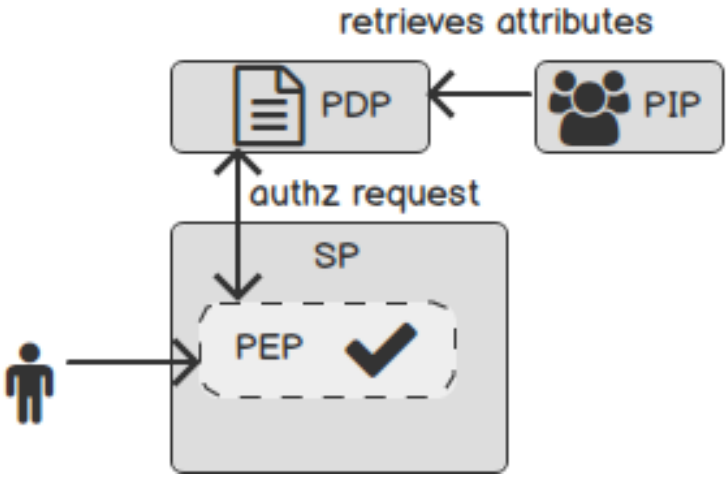

Figure 3: Reference architecture of XACML

2.1.2 Federated Identity Management

In Federated Identity Management (FIDM), SPs from different security domains exchange messages containing authentication and authorisation credentials about users [1]. As a result, federated IDM reduces the cost for SPs because they no longer manage users that are not under their control (i.e., user management may be under the control of the partner service) [2]. Figure 4 shows SPs from different security domains may form a federation by developing an offline operating agreement [18].

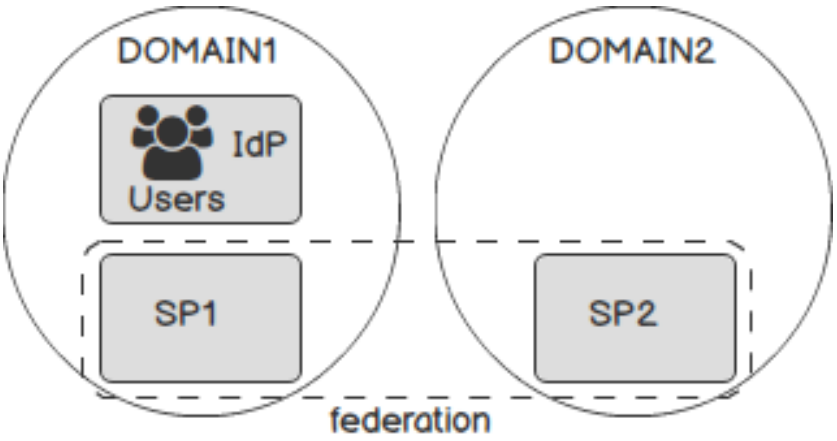

Figure 4: Federated Identity Management

In the offline operating agreement, SP1 and SP2 may specify which PII they want to exchange, which protocol to use (e.g., Security Assertion Markup Language (SAML) [19], or Web Services Federation [20]), and whether SP1 may act on behalf of a user to access SP2. In the end, SPs expect each other to behave accordingly and thus they establish a trust relationship [18]. Afterwards, developers adapt the implementation of Authentication and Authorisation Infrastructure (AAI), depending on these agreements [2].

2.2 Motivation for Identity-as-a-Service

The cloud computing reference architecture [21] defines three main roles in any cloud computing environment: a *service creator* develops a cloud service running on one or more platforms, a *cloud provider* who runs those services on demand, and *cloud consumers* or end-users who consume these services. Based on these roles, the research defines the following scenarios:

2.2.1 Single-Sign-On

In Figure 5, a delivery service is hosted in a cloud provider, and a shopping service migrates from a local hosting to it. The cloud provider already has a number of users (e.g., a user Bob), who consume the two services.

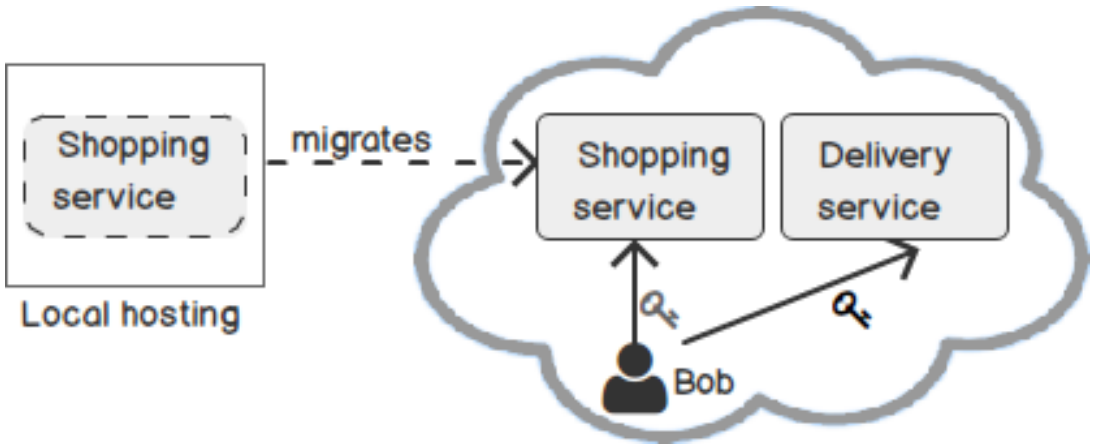

Figure 5: Shopping application migrates from on-premise to a cloud hosting

Without Single-Sign-On (SSO) support, Bob has two credentials to authenticate to each service separately. To give Bob an SSO experience, the shopping service has to adapt its AAI implementation to the IDM system (of the cloud provider). The adaptation requires security mechanisms protecting the exposed component against external users from the cloud. This scenario raises a question: how is it possible to adapt the AAI implementations of cloud services in a target IDM system on demand?

2.2.2 Business-to-business integration

In the second scenario, the cloud provider may have a statistic report about the business market place. The report may indicate that most users, who use the shopping service, also use the delivery service. Based on such a statistic report, the creator of the shopping service may want to cooperate with the delivery service on the same hosting environment as its service backend as in Figure 6.

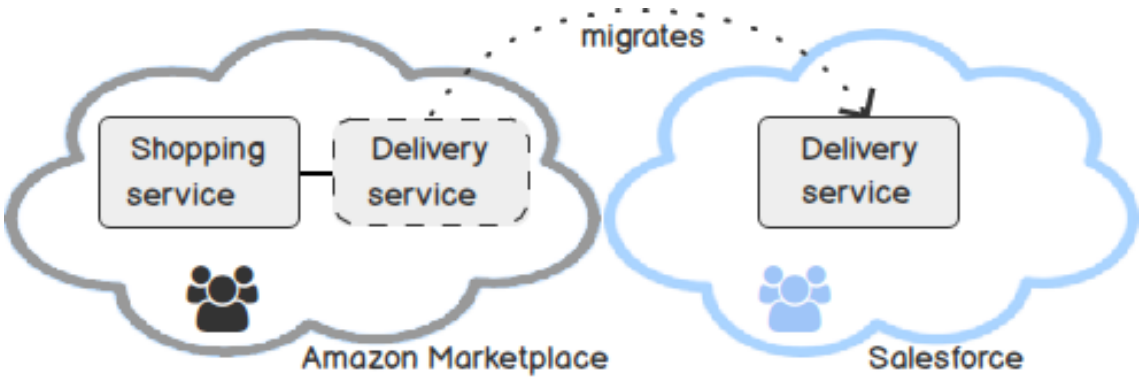

Figure 6: Shipping service migrates from Amazon to Salesforce

In the beginning, they may adapt their AAI implementations manually. When the delivery service migrates to another cloud provider that may have a different IDM system, the two services have to adapt their AAI implementations again. As an Independent Service Vendor running on many platforms, the service creators may refuse to change their implementations to adapt to a given cloud service. The statistic report may be useful in the current business marketplace, but in another market, the shopping service may cooperate with a different service. This scenario raises a question: how is it possible to adapt the AAI implementation of cloud services so that they can update and terminate their B2B relationship on demand?

2.2.3 Dissemination of user identity

In recent years, users may store their PII in the cloud environment so that cloud services may access and use it on demand. In Figure 7 (on the left), Facebook is a public IdP that collects PII about users. After a user authenticates to Facebook, he can access an application in this domain (step 1 and 2). Facebook may also disseminate user identity to the application on demand (step 3). According to the Facebook data scandal in early 2018 [22], an application was allowed to collect PII of 50 million users for “academic” use but gave the collected data further to a company, Cambridge Analytica, for “analysis” purpose. This example shows that users typically disclose their

identities with an SP (e.g., SP1) over the frontend. In the backend, SP1 may consume another service (e.g., SP2) in a B2B relationship (step 4). SP1 may be dishonest or accidentally forward PII to SP2 without user control. This scenario raises a question: how is it possible to support identity propagation between intermediaries but also protect unauthorised access at the same time?

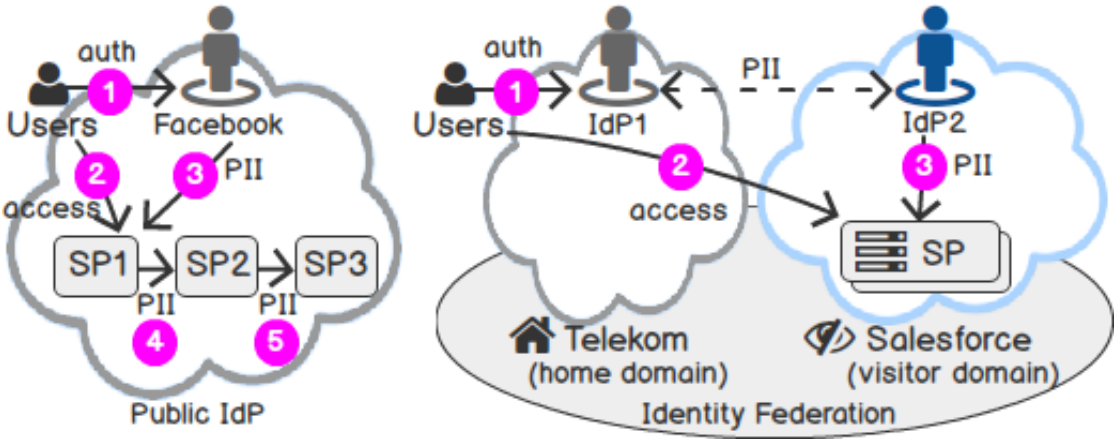

Figure 7: A user privacy scenario in SaaS cloud provider[1]

In the last scenario (Figure 7, on the right), a company (e.g., Deutsche Telekom) offers its employees several cloud services hosted by a cloud provider (e.g., Salesforce). The applications on Salesforce may require user identities exclusively. Thus, it may be necessary to provision some user identities (e.g., home address, tax identification number) from Telekom to Salesforce, while keeping the authentication credential (e.g., username and password) locally at Telekom. After a user authenticates successfully at Telekom (step 1), he accesses the applications at Salesforce (step 2). This is a typical solution for a local company that uses an SaaS [23].

Thanks to identity federation, cloud services can query user identities from the IdP of Salesforce without contacting Telekom and gain performance (step 3).

[1] Note: This thesis uses Salesforce, Facebook, and Telekom in the examples since they represent familiar examples, but the examples are not tied specifically to these organisations and the solution concepts are equally applicable to other vendors and services.

However, the disadvantage is that Telekom must completely trust and delegate the control for disclosing its employees' data to Salesforce. This scenario raises a question: how is it possible to protect PII against an *honest-but-curious* entity in identity federation? It means Salesforce may follow the protocol and computation that Telekom delegates to it, but may try to learn more information about the stored data.

Even if the two companies specify privacy policies in their business contract, it is not possible to prevent a malicious host, malware, or an insider attack. Indeed, attackers have stolen 1.5 million personal data from the health authority in 2018 because the IdP had malware injected in it [24]. This scenario raises a question: how is it possible to protect PII against an *untrusted host*?

2.3 Research Challenges

In e-commerce, SPs demand highly secure and flexible access control mechanisms for identity federation [3]. However, this security feature is not a core competency. Therefore, SPs may prefer outsourcing AAI to a third party so that they can focus on developing their core business functionalities. Work in [3] defined four levels of outsourcing AAI as in Figure 8:

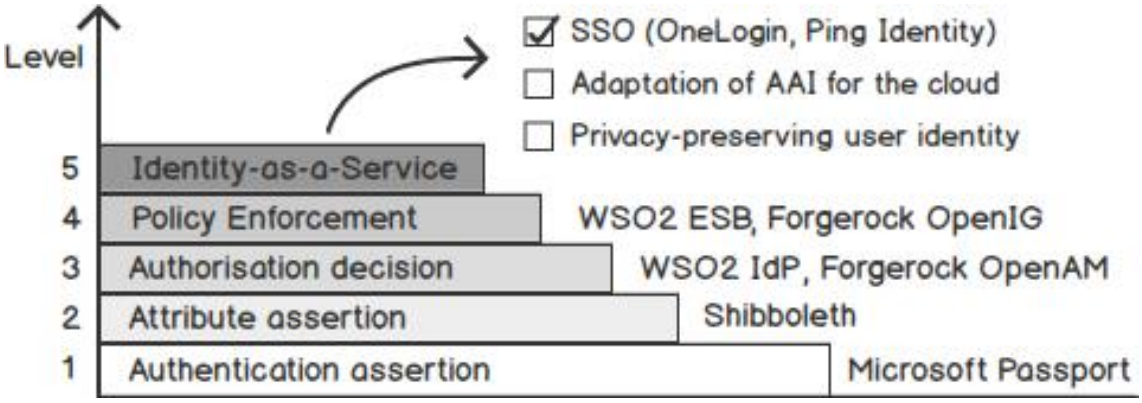

Figure 8: Authentication and Authorisation Infrastructure levels

In level one, SPs use a central authentication service. In level two, user attributes are stored in a central place and disseminated to SPs on demand. In

level three, remote service stores policies for access control. Finally, the last level provides a proxy to enforce access control's decision and protect an application in the backend. In this thesis, IDaaS proposes the fifth level of outsourcing AAI with two novel requirements: IDaaS adapts AAI for cloud services to complete a B2B transaction on-demand and protects the user identity in federated domains. To achieve this goal, the research identifies a number of challenges, as discussed in the sub-sections that follow.

2.3.1 Challenges in cloud adaptation

Work in [8] categorised four migration types to adapt an application for the cloud environment:

1. A local component of an on-premise application is *replaced* by a cloud offering (e.g., a local database is replaced by a Database-as-a-Service).
2. Components of an on-premise application *partially migrate* to a cloud provider.
3. The whole application migrates to a cloud provider.
4. The application runs on the cloud by composing with other cloud services.

The migration type 3 has the least adaptation effort because the structure of an application does not change: all components running on a physical machine now run on a Virtual Machine (VM) with the same Operating System. The adaptation may focus mainly on the configuration parameters of the hosting environment.

Except for type 3, the adaptation for the other types requires substantial implementation changes as follows. Figure 9 shows the migration for type 2, whereby the application components from the same organisation (e.g., SP1 and SP2) are distributed in various environments: a local hosting and a cloud

provider. The functionalities of the component SP2 are exposed to cloud users or users from a partner company. This migration type separates public functionalities from internal ones for enforcing different security requirements and for scalability [8]. Therefore, the adaptation requires security mechanisms protecting the exposed component against both internal and external users.

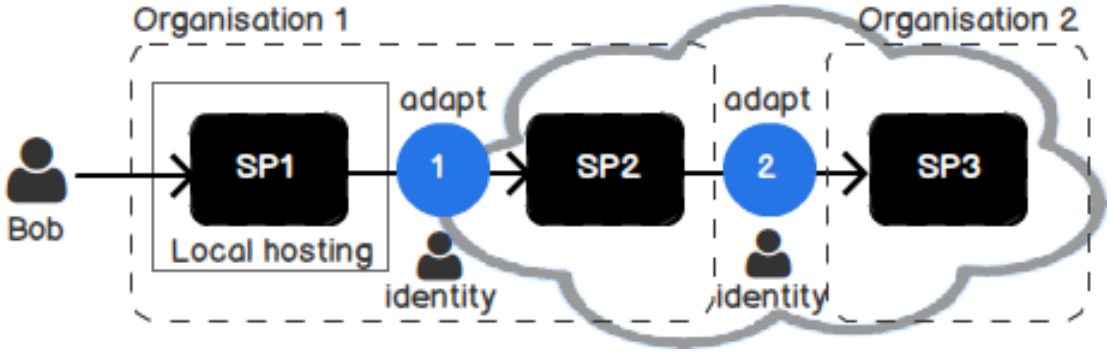

Figure 9: Security adaptation for a business-to-business transaction

Figure 9 also shows the adaptation for type 4 between two cloud services (e.g., SP2 and SP3) from different organisations. In this type, the adaptation requires security mechanisms protecting the exposed components from different security domains.

In general, IDaaS could adapt the AAI implementation so that SPs from various security domains can trust each other as well as update, and terminate the relationship upon each change. Here the AAI adaptation may happen not only between external applications but also between the internal application components of the same application.

Furthermore, cloud adaptation typically faces two obstacles of *portability* and *interoperability* [25]. Portability is the ability to move an application from one cloud provider to another one at the lowest possible cost, effort, and time [25]. Therefore, the adaptation of the required security infrastructures should be portable in multiple cloud providers together with the application components that it protects.

Interoperability is the ability of two or more components to exchange information and to use the exchanged information [25]. In order to complete a business transaction, SPs in the call chain (e.g., SP1, SP2, and SP3) may exchange information about the original caller (e.g., user Bob). However, an intermediate service (e.g., SP2) is like a black box implementation because only its developers can understand the implementation and propagate the user identity from a service consumer (e.g., SP1) to a partner service (SP3). Therefore, IDaaS should (somehow) understand the intermediate services and adapt the required security infrastructures to propagate the user identity from the original caller to the end SP.

2.3.2 Challenges in privacy-preserving user identity

From the user's perspective, users register to an IDaaS of their choices that not only enables them to access various cloud services but also preserves their privacy. From the motivation scenario, IDaaS may protect PII over an *intended channel* (i.e., a frontend service), as well as an *unintended channel*, against an *honest-but-curious* and *dishonest* partner service in FIDM, as well as an insider attack, and an *untrusted host*.

1. **An unintended channel**: It happens when users may interactively disclose their PII to a front-end service. However, to complete a B2B transaction, SPs may propagate PII over intermediaries in the call chain. In such a case, the intermediary service is completely transparent to the users. Therefore, users cannot control the disclosure of PII to a back-end service. The challenge is to allow users to control the disclosure of PII over intermediary services without knowing their services' identities explicitly. It means users may not know the SPs in the call chain beforehand when they disclose their information.

2. **An honest-but-curious SPs or IdP**: When two or more organisations federate and share their IDM systems, one organisation may follow the protocol and the business contract but may try to learn more information about the users of the other ones. The challenge is to prevent a partner organisation or SPs in the call chain to learn more information than what they are allowed to do completely.
3. **A dishonest SPs or IdP**: When two or more organisations federate and share their IDM systems, one organisation or SPs in the call chain may not follow the protocol. In such a case, the challenge is to identify the dishonest entity so that it has to pay for the penalty.
4. **An insider attack**: When two or more organisations federate and share their IDM, an employee from one organisation, who is in a potential position having access to the IdP or the database (e.g., an admin), may try to steal users information from the other organisations. In such a case, the challenge is to prevent the adversary from one organisation to access user data from the other ones.
5. **An untrusted host or a malicious host**: It happens when the host may deploy malicious software on the server. Since the host has full control over its execution and fully understands the program, it may change the executions of the program [26]. In such a case, the malicious host may change the executions of the authorisation service, which control access to the stored PII on the host, to bypass the disclosure policies. For example, a system may allow a user with the “doctor” role to access information of his patients under treatment. However, the host may change the executions to allow any users to access this information. Because ones cannot guarantee all hosts in the federated domains are trusted and not malicious, the thesis may assume

all hosts are untrusted by default and protect PII while storing PII on these hosts.

2.3.3 Challenge in performance

The authorisation model ABAC has the following limitation in performance [27]: For each user request to a resource, a PDP further queries user attributes from a PIP for an authorisation decision. If user attributes are stored in an external location or distributed in federated domains, the PIP has the latency to collect and synchronise user attributes. As a result, the authorisation decision has a delay for each user request or fails to make a decision when the required PII is not available.

In the proposed reference architecture of IDaaS, the evaluation of an authorisation request should be *computation autonomy* and *data autonomy* if possible. Computation autonomy means, the PDP has the entire codes for its computation and does not involve multi-party computation. Data autonomy means, the PDP uses the information that it possesses during the computation but does not receive any external data from another party. These requirements reduce the network interaction between multiple entities for evaluating an authorisation request.

2.4 Conclusion

FIDM enables SPs to integrate with each other based on shared Identity Management. However, FIDM also comes with new challenges, especially in the cloud environment. On one hand, migration of existing applications between environments requires substantial adaptation effort in the AAI implementation so that the implementation is portable and interoperable together with the application components that it protects. On the other hand, FIDM disseminates

the user identity over intermediaries in various domains. This chapter has identified five challenges in protecting PII with consideration of the performance issue while propagating PII in distributed systems.

Responding to the first challenge of cloud adaptation, researchers have proposed several approaches as discussed in the next chapter.

Chapter 3

State-of-the-art in AAI adaptation for the cloud environment

Chapter 2 has outlined three research challenges. One of them is the adaptation of the security infrastructure for cloud applications. This chapter continues to review working effort that addresses this challenge. Section 3.1 first reviews the current design patterns of IDM and explains the missing gap in these architecture designs. Then Section 3.2.1 and Section 3.2.2 review the literature for the adaptation effort with functional and security aspects for cloud applications, respectively. Finally, Section 3.3 concludes the missing gap in the research literature in general.

3.1 Identity Management design patterns

Managing identities and provisioning them in systems are problems that information security has been tackling for decades [28]. One difficult decision of IT administrators is to choose a suitable architecture for their organisations [28]. To tackle this issue, several surveys [18, 28, 29] have collected possible design patterns as in Figure 10: a centralised, a free-form model, a circle of trust, an identity broker, an isolated, and a user-centric IDM.

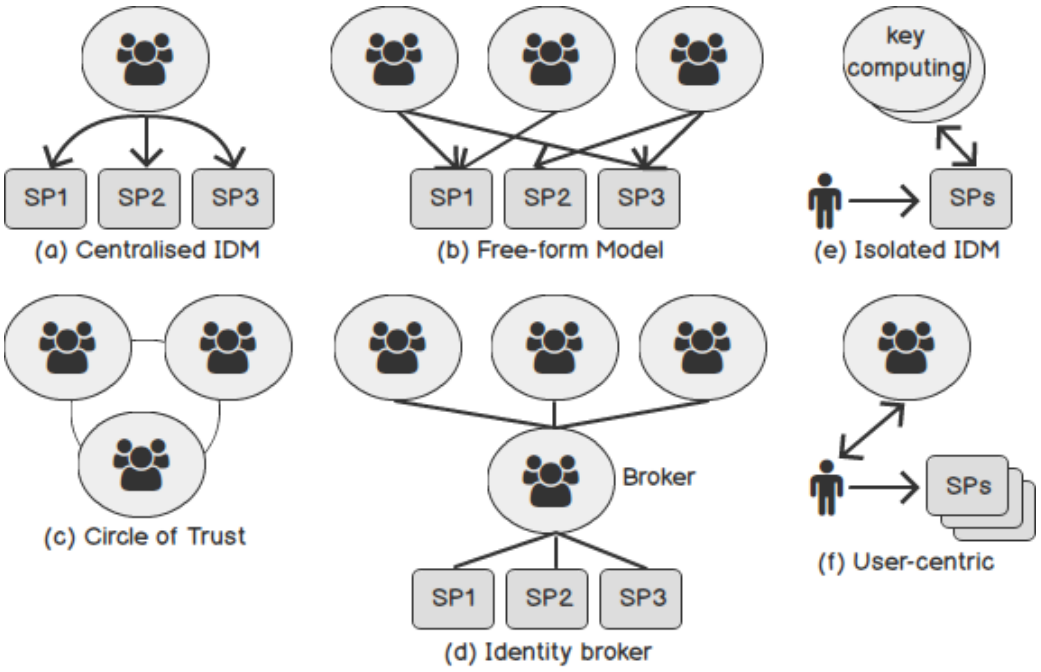

Figure 10: Identity Management design patterns

3.1.1 Centralized IDM

A central IdP stores the user identities and distributes them to SPs within one security domain as in Figure 10a. Microsoft Passport [30] is an example of this design pattern. This solution has a limitation that all SPs have to trust a single service for authentication.

3.1.2 A free-form model

To solve the limitation of a single authentication service, the free-form model allows SPs to connect directly to multiple IdPs as in Figure 10b. This solution has a limitation that all SPs have to adapt the implementations with various IdPs at the same time. As a result, an identity federation to an external provider may be complex and technically difficult [28].

3.1.3 Circle of Trust (CoT)

In contrast to the centralised IDM, the Circle of Trust (CoT) [18] requires all members completely trusting each other to exchange information about their subjects as in Figure 10c. As a result, users may authenticate to an IdP of one SP and consume all other services in this CoT. However, all SPs exchange user attributes with no filtering and protections. Therefore, this pattern has the least control over user privacy. An existing implementation of this pattern is the Liberty ID-FF [31].

3.1.4 Identity broker

In the security guidance for cloud computing [28], the Cloud Security Alliance recommends using an identity broker as an IdP in the middle as in Figure 10d. All internal IdPs from different parties communicate with the central identity broker, which again serves as the IdP for the federation to multiple SPs. In

comparison to the free-form model, the identity broker prevents the communication silos and simplifies the implementation for the SPs. The internal IdPs can federate a portion of an organization's identities through the identity broker as well [28]. However, this design pattern does not clarify: who is accountable for identity theft. The identity broker does not belong to any security domains. It does not belong to the domain of a user organisation, a cloud provider, or a cloud service. In practices, security guidance [28] mentioned that an external company may provide the identity broker as a service. In such a case, all internal IdPs federate and transfer PII to the single identity broker. Individual users (or a company), who use the identity broker as the service in the middle, cannot control the disclosure of PII over multiple parties behind it. In the case of identity theft, they cannot identify the responsibility of any parties behind the identity broker.

3.1.5 Isolated IDM

In contrast to the centralised IDM, CoT, and identity broker, the Isolated IDM does not rely on a Trusted Third Party (TTP) for issuing and verifying credentials because the TTP could be an untrusted host. Work in [32] gives the users the possibility to issue their credentials (in an encrypted form) and give them to SPs directly. The SPs can evaluate the predicate of the encrypted credentials by contacting multi-party computing as in Figure 10e. It does not mention how the users can prove that they possess the attributes because without a TTP the users can only self-assert their attributes. The major limitation of such an approach is scalability when the numbers of users and cloud services in the cloud environment increase.

3.1.6 User-centric IDM

This design pattern involves a user as a man in the middle to retrieve credentials from a central IdP in the first phase. In the second phase, the user selectively decides which credentials he wants to prove to an SP (see Figure 10f). Because the two phases happen asynchronously, the users avoid direct contact between the IdP and the SP.

Discussion: The taxonomy survey in Inter-Cloud [33] denotes *federated cloud* for cloud providers who voluntarily share their resources such as academic research projects. The taxonomy also denotes *multi-cloud* for independent vendors who do not share or lease resources despite their data center outages such as Amazon EC2 and Microsoft Azure. According to this taxonomy, the CoT pattern may fit federated cloud because cloud providers volunteer to share their resources, and academic research projects may get the most benefits from it. On the other hand, the identity broker may fit *multi-cloud* because this pattern federates user attributes between security domains. However, it relies on a single broker to be accountable and responsible for many parties. Finally, the user-centric approach overloads the IDM tasks to the user's decision in every transaction [34]. The "seven laws of Identity" stated that human-PC link is the weakest link compared to the links between the PC, SPs, and IdPs. Therefore, the human link is the major target of Identity theft [1]. Also, the user-centric approach does not support multiple security domains as the identity broker.

3.2 Adaptation effort for the cloud environment

Adaptation of existing applications for the cloud environment requires substantial implementation changes in individual architectural components. A survey from 2012 to 2017 [35] has identified 91 research efforts for cloud

adaptation, but very few papers have addressed the security aspects. In the following sections, the research reviews the adaptation effort with the functional and security aspects in Section 3.2.1 and Section 3.2.2, respectively.

3.2.1 Adaptation effort with functional aspects

Although the following section does not directly address the research challenge in IDM, it helps researchers to understand the common challenges and solution efforts to the cloud adaptations in general. This information is useful because, in the end, IDaaS is also an adaptation effort of an application with the security aspects for the cloud environment.

In the migration of type 1, an application component (e.g., a local database) is replaced by a cloud offering (e.g., a DynamoDB [36]). Developers may read the source codes of their application, search for all relevant codes, and change the implementation to use the DynamoDB manually. Such adaptation tasks for the Data Access Layer are repetitive in the same application and could be reusable in another one [37]. For this reason, work in [37] considers the adaptation process as repetitive tasks and can be reusable. They support developers by defining “search and replace by template” patterns to reduce the adaptation efforts of developers from a manual modification by 60% – 90%. However, developers require a deep understanding of the implementation to annotate specific blocks of their codes for the transformation. Therefore, this approach is neither scalable nor holistic.

On the other hand, the mOSAIC project [38] considers cloud offerings from multiple cloud providers as programming interfaces. Because all resources of the same type should possibly have the same interfaces, their approach provides a unified API that wraps the native APIs of all cloud offerings. For

instance, it provides a unified API for provisioning a distributed database or a message queue for communication between the application components. At the design time, developers split their application into components and use the unified APIs to wire the application components with each other. As a result, developers compose their application components using the best cloud offerings. In an ideal case, the framework should transform the programming language, which the developers use in their applications, to a programing language of the native cloud offerings with a little overhead. However, this approach also introduces a potential loss of native features that the cloud offerings provide [39].

Migrating an application to the cloud may affect all application layers, including the data layer and business process layer [8]. However, most enterprise applications have no clear picture of their topology [8]. The more complex the system, the more effort and error-proneness is involved in adaptation upon topology changes [8]. Therefore, work in [40-43] consider the migration as a *reverse engineering* process that supports developers to understand their software system better, then to upgrade, and to reuse parts of it in another system. They use the framework MoDisco [44] to generates a *topology view* of a given application by extracting relevant information from source codes and configuration files of the application. As a result, they decompose a legacy application in a topology view (Figure 11, phase 1).

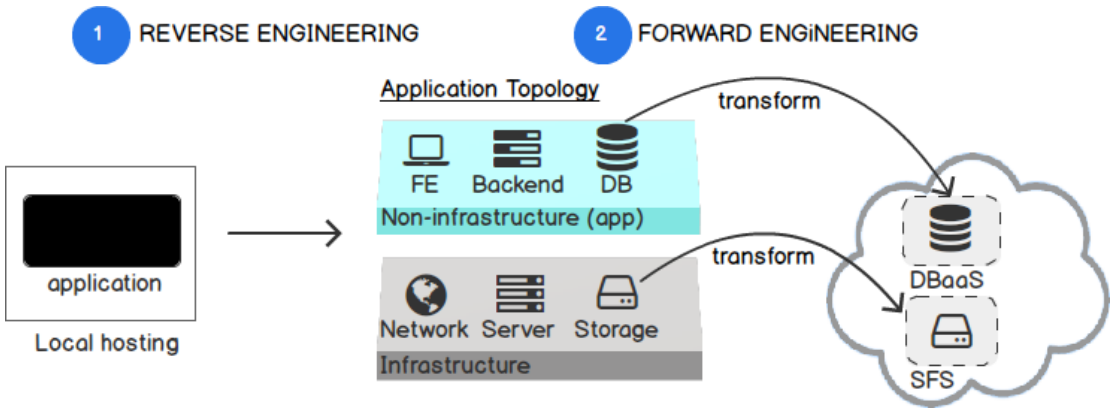

Figure 11: Phases in cloud migration (reverse engineering and forward engineering)

A topology is a directed graph that describes all application components (with deployable artefacts) and the relationships between them (i.e., hosted-on, depends-on, connects-to) [41]. Then a *forward engineering* process identifies a sub-graph of the topology that includes all relevant components to migrate to the cloud. The project ARTIST [42] enables matching components in the view with various cloud offerings and provides a transformation process to replace them.

Another European project MODAClouds [45] supports the provisioning and adaptation of applications in multiple cloud providers. In their approach, the applications are designed with the goal to be portable in the cloud environment from the very beginning. Therefore, they are not necessarily reverse-engineered. At design time, developers describe the topology of the cloud application in a *Cloud Provider Independent Model (CPIM)*, and at run-time, an engine reads the CPIM description and provisions the resources in a target cloud provider. In the implementation, the engine uses the jClouds framework [46], which provides an intermediate layer for calling APIs of multiple cloud providers and provision resources in them.

On the other hand, an adaptation process requires evaluating the deployment of an application component in a target hosting. In general, a Trusted Third Party

may assert certifications for the deployment by using a Trusted Platform Module (TPM), a monitoring-based, or a test-based certification [47]. For instance, work in [48] uses TPM to handle a key exchange and bootstraps an application on a remote machine. The bootstrap application confirms the machine's status before any application components are deployed on the machine. For a monitoring-based certification, work in [49] collects security events and reports back to the central monitoring for analysis. For a test-based certification, work in [50] describes "policy" annotations in the application topology. Before the deployment, administrators can choose which policy they want to apply and an orchestration engine will call the corresponding execution plan.

3.2.2 Adaptation effort with security aspects

The following work adapts an application for the cloud environment with security aspects. They are organised in two categories: protection on the network layer and on the application layer.

3.2.2.1 Protection on the network layer

The Cloud Security Alliance defines Security-as-a-Service (SecaaS) as a third-party that provides security capabilities (e.g., security assessment, intrusion detection, Web application firewall etc.) as a cloud service [28]. In this context, one may think to forward all network traffic to an external SecaaS provider before allowing it to reach to the destination service as in [51]. To implement SecaaS, they use Unified Threats Management (UTM) that inspects network packets against all possible threats. From their results, the performance for HTTP and database traffic (even with a medium load) reduces significantly. The response time is low because all traffic is forwarded to the UTM provider and then delivered to each application component [51]. Most importantly, a user

company has to maintain two Service Level Agreements: one with the UTM provider and one with the cloud service. When a security incident occurs, the user company may have difficulties to identify, who is accountable [51]. Finally, the UTM can only protect the front-end services from external threats.

Sun *et al.* [52] go further to protect the application components from both external and internal traffic. They add a network flow on virtual devices (e.g., the virtual bridge br-int in OpenStack [53]) to forward a network packet to a security VM before it reaches an application VM. As a result, their approach adds more flexibility to control the internal traffic between VMs in a private network with minimal overhead (e.g., to forward all packets with TCP protocol on port 80 to a Web application firewall VM). However, it requires a cloud provider support by modifying the Software Defined Networking implementation and adding an additional security VM on each compute node (to reduce the network traffic between the security VM and the application VMs over the tunnelling network between the compute nodes).

3.2.2.2 Protection on the application layer

On the other hand, a Gartner report [54] recommends a different design: They do not recommend to forward all network traffic to a centralised security service but to embed a security service within the application components. In such a case, a security solution spreads across all application components. Also, the security implementation is visible to the SP for audit and control. Thus, SP can be accountable in case of incidents. In comparison to Sun *et al.* [52], this design also reduces the costs of introducing an additional security VM.

The following approaches follow Gartner's recommended design. Work in [55] uses Aspect-Oriented Programming (AOP) to enforce authentication and

authorisation on critical points of an application program. In particular, it uses the tool Yiihaw [56], which modifies binary files of an application to intercept the execution of a component, a class, or a class method, and redirect a client request to a central Policy Enforcement Point (PEP). This approach also provides a Web interface for administrators to specify security policies in a central management system. However, AOP requires programmers to read the code and understand what is happening in order to prevent errors [57]. For instance, to read the input parameters from the target method, an advice method in Yiihaw [56] needs to define the same parameters ordering as the target method. Therefore, administrators may have difficulties to define security policies by looking at the class methods. In addition, this approach adds additional performance overhead on the secured resources [55].

On the other hand, the GEYSERS project [58] use a common Enterprise Service Bus (ESB) [59] and WS-Trust [60] to establish trust and enforce authorisation within cloud services. This solution perfectly fits in an academic research project, whereby organisations volunteer to use the same framework implementation. However, no global standards exist for ESB concepts or implementations [59] (i.e., each ESB vendor defines their messaging encapsulation differently). The aim of this research is to adapt AAI for cloud services with no tight binding in a specific vendor implementation. Therefore, the research cannot consider a common ESB as a solution.

Another project OpenPMF [49] focuses on the management of security policies. It generates the policies (i.e., from a model-driven language to a machine-enforceable policy) and pushes the policies feeds to a dedicated PEP of a target application. This approach still requires a solution that provisions the PEP in multiple cloud providers and interoperable with the application in the backend.

3.3 Conclusion

This chapter reviewed the current design patterns of IDM. None of the architecture designs is suitable for disseminating PII in federated domains and protecting the confidentiality of PII from unauthorised disclosure at the same time. Therefore, the thesis may propose a better architecture design for IDaaS that does not rely on a single identity broker and each party involved is accountable for the information they disclosed in a controlled way.

The research also reviewed solutions that adapt the security infrastructures for cloud applications. While some of them use a central service to enforce protection for multiple cloud services on the network or the application layer, the other ones implement dedicate protection in front of each application component. Both designs have pros and cons. However, they do not provide a flexible solution for cloud administrators to choose at the deployment time. In practice, administrators may choose a central service to protect their application components in one cloud provider but spread a security solution across application components in another one. In comparison to them, IDaaS provides this flexibility and portability for the security infrastructure.

Furthermore, all approaches so far focus on protecting cloud services against possible threats. None of them can govern the identity propagation from the original caller to the end SP to complete a business transaction on demand (i.e., the interoperability issue of the security infrastructure). Also, none of them protects PII across SPs in the call chain. In comparison to them, IDaaS considers both aspects (i.e., the interoperability and PII protection) in one solution.

Among these approaches, it is worth mentioning that the AOP approach draws the researchers' attention as it enforces authentication and authorisation on critical points of an application program but slows down the application performance. It motivates researchers to search for an alternative solution that enforces authentication and authorisation but does not affect the application performance. Chapter 6 will continue these findings.

Apart from this, related work on adaptation with functional aspects introduces a holistic approach that decomposes a legacy application in a topology view. They consider a cloud adaptation as a reserve engineering process that transforms a "black box" application to a topology view, from which an application component can be further adapted in a target environment. The thesis may borrow the idea of a topology view as a holistic solution to adapt the security infrastructures. The test-based certification also inspires researchers to model the security requirements with some default integration tests so that the adaptation process can evaluate any changes in the runtime environment to conform to the desired model.

Chapter 4

State-of-the-art in privacy-preserving user identity

Chapter 2 has outlined three research challenges. One of them is privacy-preserving user identity. This chapter continues to review working effort that addresses this challenge. Section 4.1 first explains the properties of user privacy. Then Section 4.2.2.1 and Section 4.2.3 review two approaches that preserve these properties: the *anonymity* approaches and *confidentiality* approaches, respectively. Researchers also explain why they did not choose the anonymity approaches in their solution. Section 4.3 reviews related work of *Purpose-based Access Control* and explains why researchers choose it to control the disclosure of PII. Moreover, Section 4.4 reviews *secure computation* and explain why researchers cannot use these solutions. Finally, Section 4.5 gives a literature review on *Functional Encryption* and explains why researchers choose it to protect PII from untrusted hosts.

4.1 Privacy properties

The privacy-driven approach has the following privacy properties [61]:

1. *Undetectability*: When SPs interacts with an IdP to obtain authorisation credentials about a user, the IdP can detect the authorisation request has occurred on the SP and observe the requested user attributes and user behaviour. Undetectability is a privacy property to prevent adversaries from inside the IdP to collect this information. If users trust the IdP for not revealing information about their actions, then hiding authorisation requests from the IdP is not necessary [61].
2. *Unlinkability*: This privacy property concerns, whether various SPs can associate multiple transactions with the same user. Unlinkability prevents one or more adversaries from SPs to collude and aggregate attributes about a given user. This property is important to protect the *anonymity* of a user in the following way. In a given domain, a user is considered as *partial*

anonymous within a set of subjects if this set is large enough and the subjects potentially have the same attributes [62]. However, if adversaries can aggregate attributes about a user from different sources, the set becomes smaller for adversaries to identify the user [62]. *Untraceability* is a special case of unlinkability that even if the IdP and the SPs collude with each other, they cannot link which authorisation credential (issued by the IdP) is used for which authorisation request of the SP. In contrast to untraceability, *accountability* is a desired function to reveal a user identity when a user abuses his anonymity. In such cases, a trusted party revokes the user identifier so that the user is responsible for his action. If users trust SPs for linking their actions in a transaction, then unlinkability is not necessary [61].

3. *Confidentiality*: This privacy property ensures that user identity is accessible only to those who have authorisation. Confidentiality is a design goal to protect data when it is stored and disseminated in distributed systems [63].

Complete anonymity (i.e., undetectability and unlinkability) would be equivalent to perfect privacy [63]. Therefore, operating under anonymity is one way to achieve privacy [63]. However, in many cases of e-commerce, SPs may require users to reveal sensitive information to complete a business transaction. In such cases, it is possible to sacrifice user anonymity and protect the confidentiality of disseminated data [63]. In the past, several approaches followed one of these two protection goals as described below.

4.2 Privacy-preserving user identity

This section organises related work into three categories: standard approaches, anonymity, and confidentiality.

4.2.1 Standard approach

Recently, the European Union (EU) have defined the General Data Protection Regulation (GDPR) [15]. It is a lawful regulation for a data controller (i.e., an organisation that collects data from EU residents) and a data processor (i.e., an organisation that processes data). Briefly, the collection of personal data should be lawful under the consent of users and with a specified purpose for which they are used. The purpose of data collection has a time limit (from the time to collect data to the time to fulfil the purpose). For later use, PII should not be disclosed for other purposes than the ones they have been collected. While this regulation does protect the privacy of individuals by law, it still has the following limitation in implementation. For instance, the consent receipt specification [64] defines a record of authority when a user discloses his PII to an SP. However, this approach involves the user granting permissions for each SP explicitly.

4.2.2 Anonymity approach

In the traditional X.509 PKI signature [65], a user typically obtains an authorisation credential issued by a trusted party and sends it to an SP. The credential may contain the user's public key and user attributes in plain text that is verifiable by the SP. This approach has three disadvantages as follows. First, the user reveals all attributes in the credential to the SP. Second, the SP may reuse the credential to impersonate the user to a third party (as long as the signature is still valid and has no audience restrictions) [66]. Third, because all requests are verifiable by the user's public key, different SPs may collude each other to link all requests to the same user (i.e., linkability) [34]. In the following section reviews various approaches that solve the issues above. In general, they use the Zero-knowledge Proof (ZKP) protocol [67] to convince an SP that a

user truly possesses a certain attribute. The SP may learn a user attribute but cannot prove it to another party.

4.2.2.1 Anonymous credentials

In anonymous credentials [10] [68-70], a user receives a master credential from a trusted issuer in phase one. The master credential contains all user attributes signed by the issuer. Then in phase two, the user uses his master secret key to generate a *pseudonym* and a subset of credentials from the master credential, and convince an SP that trusts the issuer.

Figure 12 shows that a user Bob can show an exact attribute value (e.g., name="Alice"), comparative arithmetic expressions (e.g., age>18), and logical expressions (e.g., country="DE" OR country="UK") to various SPs.

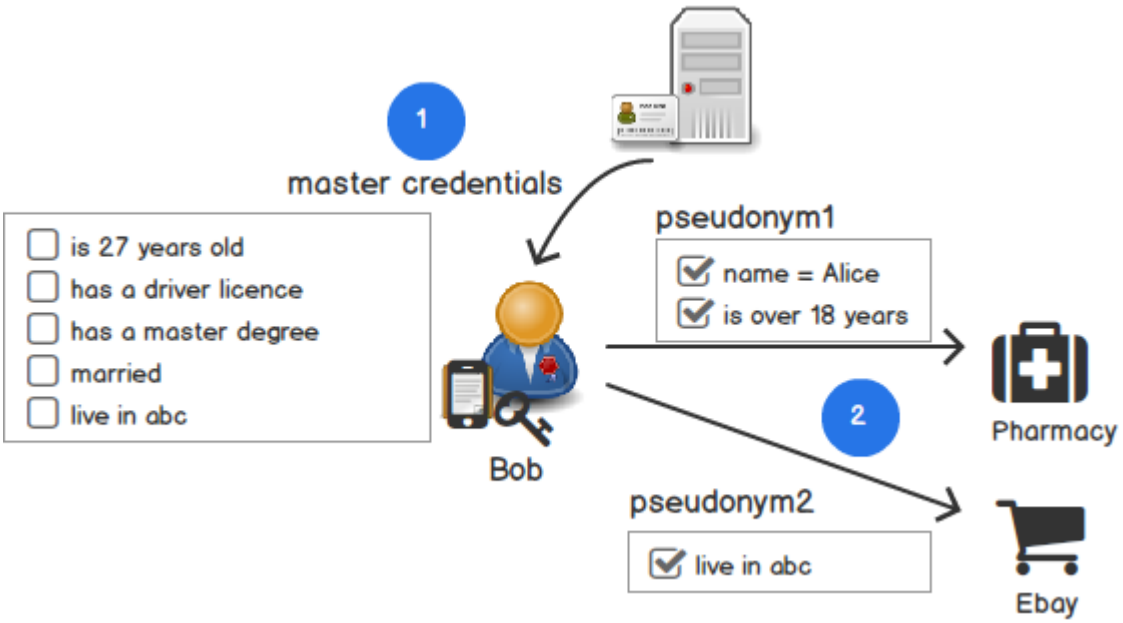

Figure 12: Anonymous credentials

By showing an exact attribute value, Bob can selectively disclose an attribute from the master credential in a transaction. By using comparative and logical expressions, Bob can protect the disclosure of his attributes to an SP. In the example above, the SP does not know exactly the country, where the user comes from. Most importantly, the SP cannot reuse the credential in another

transaction to impersonate the user, because the SP does not possess the user's master secret key. Furthermore, by using different pseudonyms in each transaction, SPs cannot link all transactions to the same user (i.e., unlinkability). Even if the issuer and the SPs cooperate with each other, they cannot link all transactions and reveal the identity of the user (i.e., untraceability). The project ABC4Trust [34] demonstrated the integration of anonymous credentials with standard protocols like WS-Trust [60], SAML [71], and OAuth [72].

4.2.2.2 Aggregated zero-knowledge proof

Anonymous credential requires a user device to have sufficient processing power for cryptographic calculation [34]. To improve the performance, Bertino *et al.* [73] provide an efficient way of performing multi-factor proofs in an *aggregated ZKP* protocol. This approach reduces the proofs of several attributes into one single ZKP. As a result, aggregated ZKP is suitable for use on lightweight devices [73]. In [74], Bertino added an additional component, the *Registrar*. This component stores a user attribute not in clear text, but in a commitment[2] together with the signature of the commitment. When a user accesses an SP, the user first retrieves the commitments and the signatures from the Registrar. Then he follows the protocol to convince the SP that he knows how to open the aggregated commitments. In case the user needs to access multiple SPs, an SP in the frontend can act as a broker service as in [75]. The broker service collects all user attributes required by the backend services so that users only need to send an aggregated ZKP to the frontend

[2] A commitment scheme is a method that allows a Prover to commit to a value *m* to a Verifier, with a chosen random value *r*. Prover can open the value *m* later by revealing *m* and *r* to the Verifier. Alternatively, Prover may use ZKP to show that he knows how to open such a commitment without revealing either *m* or *r* [67].

service. Then the frontend service forwards the proof further to each service in the backend to verify individually.

While improving the performance of ZKP and protecting identity theft from identity forwarding, aggregated ZKP has the limitation that user has to manage all attribute values in clear text and the corresponding random value *r* for each commitment. These values are secret keys known only to the user to prove in the aggregated ZKP protocol.

4.2.2.3 Group signatures

Anonymity can also be achieved using a *group signature* [76]. In Group signature, each member of the group can sign a message that is verifiable by other members without knowing who has signed the message. Only a *group manager* can generate a signing key for each member as well as reveal a member's identity when needed (i.e., traceability). Moreover, members cannot collude with each other to reveal the identity of any members (i.e., unlinkability) [76]. Work in [77] and [78] used a group signature to protect user anonymity. In their approaches, a registrar stores user attributes in cleartext. The registrar issues a *master credential* to a user by signing a message containing all user attributes. The user keeps his master credential locally. When a user accesses an SP, he may generate a new credential from the master one. In the new credential, the user may hide irrelevant attributes, append the credential with an access time, and sign the access time with his secret key. Being a member of the same group, the SP can verify the signature of the message without knowing the identity of the user.

If an SP wants to forward user attributes to subsequent SPs, it establishes a new group signature with the subsequent SPs. That is the subsequent SPs trust

the front-end service as a new group manager. The front-end service then reuses the user credential to generate a new credential and forwards it to subsequent services.

The approaches above follow *the User-centric IDM* for attribute assertion. In this design pattern, the user first obtains a credential from an IdP (phase 1). Later on, he uses it in a transaction with an SP (phase 2). Because the two phases happen asynchronously, the IdP cannot detect, which user attributes are used for which SP (i.e., undetectability) [61]. Also, SPs cannot link credentials from different transactions to the same user (i.e., unlinkability). The ZKP prevents identity theft under the philosophy that an SP may learn the attribute value but cannot prove it to another party [73].

4.2.2.4 Limitations

The anonymity approaches have the following limitations:

- *Key theft on user device:* The user's secret key is stored on user devices that must be protected from malware, broken, or stolen.
- *Missing support for multiple domains:* In group signature and anonymous credential, the user retrieves a master credential from a central issuer. With this credential, users can perform a ZKP with SPs that trust the issuer. However, the ZKP does not work with an SP in another security domain. In comparison to their approaches, IDaaS supports identity federation within security domains. Thus, it is much more scalable for the cloud environment.
- *Missing support for an unintended channel*: Users can selectively control which attributes they want to show to an SP. However, the proof is performed over the frontend channel. Thus, these approaches overload the IDM tasks to the user device for every transaction. In group signature [77], a

frontend service may become a new group manager to forward user credential to its subsequent services. This approach is not scalable because each service in a call chain has to establish a new group with its subsequent. In comparison to their approaches, IDaaS supports identity propagation between intermediaries and removes the burden of user action on each SP.

- *Missing protection against an untrusted host:* While aggregated ZKP stores user attributes as commitments in the issuer, group signature and anonymous credential store user attributes in plain text. Recall that storing user data on a remote hosting may be concerned with malicious host and insider attacks. Protecting PII on untrusted hosts is one of the research challenges of IDaaS.
- *Missing adaptation support for SPs:* The above approaches require SPs to adapt to a new protocol (e.g., ZKP, group signature). However, most SPs, which select an IdP based on how much information they can collect from the user [12], are not likely to support anonymous credentials. From this aspect, researchers have learned that a good solution for IDaaS should be adapted to the standard protocols such as SAML [79] and OAuth [72] easily.

4.2.3 Confidentiality approach

4.2.3.1 Sticky policies

In contrast to the previous category, the following approaches do not protect user anonymity but disclose PII to an entity if the disclosure policy is satisfied. Chadwick *et al.* [80] proposed an IDM for users to store *sticky policies* in the cloud. Sticky policies are disclosure policies appended to the data, wherever they are transmitted between IDM systems. Policies are bound to the transmitted data by using a digital signature to ensure their binding. The IDM then stores the data together with the sticky policies to control the information

disclosure. However, this approach assumes that cloud providers could be trusted to control access from outside but not from inside attacks.

Mont *et al.* [81] enhanced sticky policies with *Identity-based Encryption* (IBE) [82]. In IBE [82], a sender uses the identity of the receiver (in string format) to encrypt the data and sends them to the receiver. To decrypt the data, the receiver authenticates to a TTP to ask for a decryption key. The TTP uses the receiver identity to generate the decryption key. The receiver identity binds with the encrypted data because any attempts to alter the identity will make it impossible to generate the correct decryption key [81]. In Mont *et al.* [81], a user uses sticky policies as a string to encrypt the data before sending it to an SP. If the SP wants to decrypt the data, it has to interact with a TTP to providing information such as the company policies and the current platform configurations. The TTP verifies if the given information conforms to the disclosure policies and generates a decryption key for the SP. In short, the TTP acts as a tracing service to audit a disclosure event. This approach ensures that the SP understands the policies and is accountable for any actions. Furthermore, the data is encrypted on the storage server. In case it is not possible to trust an SP to be accountable, Mont *et al.* proposed to use a Trusted Platform Module (TPM)[3] on the SP. With TPM, the TTP can verify the computer platform of the SP for integrity and trustworthiness remotely before sending the decryption key to it. However, not all SPs support a TPM in their hosting platform.

[3] TPM is a security hardware component that performs encryption, decryption, and digital singing. It can be used to verify that a host BIOS is trustworthy or not. If yes, TPM trusts the Operating System loader and one can verify installed software from tampering.

In case one cannot trust a TTP for issuing a decryption key, Pearson *et al.* [83] proposed multi-party computing that jointly calculates the decryption key. The idea is to prevent one party to create a decryption key without an audit from its partner. As a result, all participants (i.e., TTP and SPs) can control each other action on the creation of the decryption keys. While this approach distributes trust among parties, it introduces complexity in the management of a shared secret in each party. For example, an SP has to manage the shared secret for each individual user. Therefore, this approach does not scale in the cloud environment when the numbers of users increase.

4.2.3.2 Active bundles

The approaches above require a TTP to authorise access to sensitive data. On the other hand, *active bundles* [84] rely on their policy engines to perform authorisation. Active bundles are containers that are inseparable from the sensitive data they protect. Briefly, a container holds its sensitive data (in an encrypted form) and disclosure policies as in Figure 13.

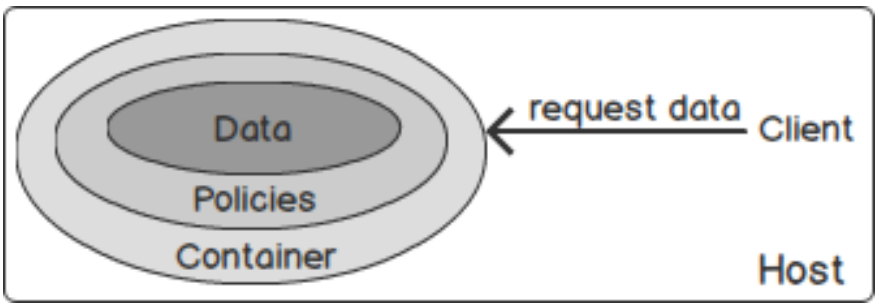

Figure 13: Active Bundles

When active bundles migrate to a target hosting, they check if the hosting environment is trustworthy and verify the integrity of their code. If all checks are proper, they enable themselves by requesting a decryption key from a TTP and decrypt the protected data. Otherwise, they will destroy themselves from an untrusted host. Afterwards, they use the disclosure policies to authorise requests from the host and disclose the sensitive data to the host. A container

could be a *mobile agent* so that it is portable within visited hosts and reduces network load for migration [84]. However, the execution of a mobile agent's plaintext code in an untrusted host is not secure [26]. For example, the host may modify the agent executions to bypass the verification process or to bypass the disclosure policies. Furthermore, the host may read the memory used by the bundle to get the decryption key (i.e., side-channel attacks) [84].

IDaaS avoids the above issues of active bundles because the cryptographic computing itself performs the authorisation. If the host changes the cryptographic computing, the decryption simply fails.

4.2.3.3 Blockchain

Recent work also took advantage of Blockchain technology in IDM. Figure 14 illustrates an authentication (or an authorisation server) that stores data in a Blockchain network.

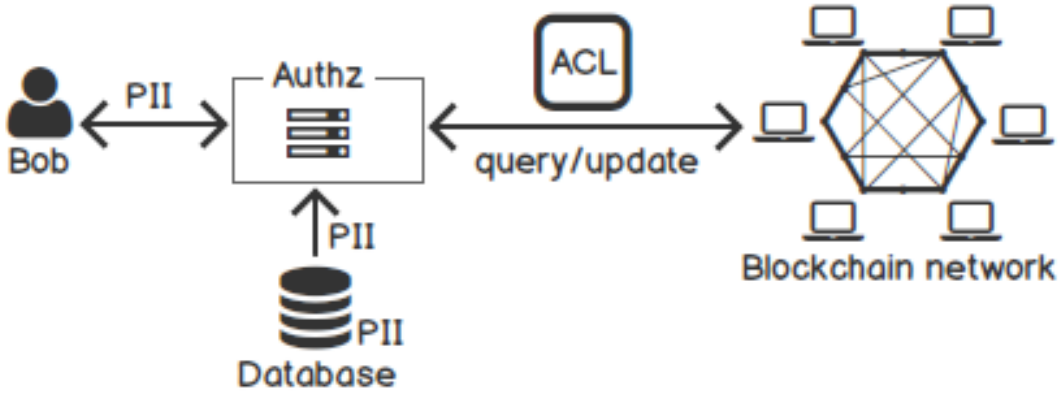

Figure 14: Identity Management with Blockchain

The stored data could be an authentication claim of a user [85] or an access control list (ACL) to personal data [86]. As a result, the stored data is replicated in all network nodes and immutable from any unauthorised modifications. However, the stored data is limited in size because it is replicated in each network node [86]. Therefore, work in [86] does not store PII in the Blockchain

but a pointer to PII in an external database. They rely on a trusted authorisation server that queries the ACL from the Blockchain and checks if the client subject has permissions to access the given PII. If yes, the authorisation server queries the given PII from the database and returns it to the client subject. While Blockchain brings an auditable history to the stored data, it cannot protect PII from an untrusted host of the database or the authorisation server. Here the approaches of using blockchain have the same issue as active bundles.

4.3 Purpose-based Access Control

Traditional access control models (e.g., RBAC) focus on which subject is performing an action on which data objects [87]. If access control is based on a certain knowledge about certain subjects (e.g., roles), it is limited to a local domain (e.g., a manager in company A can see an employee's salary, but a manager in company B is not allowed). This diversity is the main reason why the research does not consider traditional access control models but Purpose-based Access Control (PBAC) for a large distributed and heterogeneous environment.

Prior research has investigated PBAC in [88-91]. In their work, access control is rather concerned with the purposes that a data object is used rather than the actions that users perform on the data object. *Hippocratic database* [88] defines the notion of "purpose" to play a major role in access control. Data elements in the database are represented with additional columns to hold information such as purpose, recipients, authorised users, and retention. Query transactions to the database are accepted for a certain purpose. Based on this idea, researchers in [89, 90] divided purposes into two categories: *intended purpose* and *access purpose*. The intended purpose specifies the reason for which data

objects are collected. The access purpose is the purpose that a user request submits to a system for access control. A subject may have access to an object if the given access purpose corresponds to the intended purpose for which data were collected. This control makes sure that the data gathered for one purpose cannot be used for another purpose without user consent.

When storing data, the data owner specifies the intended purpose for using the data as his preference. However, the system may have difficulties in determining an access purpose of a request. Work in [89] listed the following options. First, a user may state an access purpose explicitly. However, this requires the system to trust the user completely. Second, the system may infer an access purpose based on the request context (e.g., based on the function call, the nature of the data, the application identification, and the time of the request). However, such an inference mechanism may not be accurate and not efficient [89]. So far, the research could not find any work that develops such an inference mechanism. Recent work in [92] used the RBAC model for determining an access purpose. Briefly, a user states his access purpose in a request explicitly, and the system can authorise if he possesses such purposes according to his roles in the system. As stated previously, the role-based system has limitations for the cloud environment.

4.4 Secure computation

In this section, one may assume that hosting servers are not entirely trustworthy. Therefore, the research looks for any methods to compute a function on encrypted data and provide plaintext output (of the function) to the server but nothing else. If such methods exist, one would encrypt the PII in the PIP as in Figure 15.

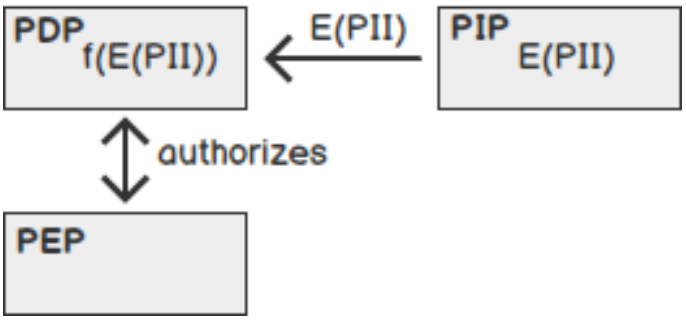

Figure 15: An idea of how to use secure computation to protect PII on an SP

During an authorisation request, the PDP would retrieve the encrypted PIP, evaluate it, and return an authorisation result to the PEP without disclosing any information about the PII to the PDP and PEP. As outlined in the research challenge in performance, the research seeks for methods that are *computation autonomy* and *data autonomy*. This requirement reduces the network interaction between multiple parties for evaluating an authorisation request.

Secure computation provides methods that allow multiple parties to compute a function over the input of each party while keeping it private [93]. The participants in the computation are considered as adversaries and not trustworthy [93]. In the following sections, the research reviews two methods in secure computation: homomorphic encryption and garbled circuits.

4.4.1 Homomorphic Encryption

In secure computing with encrypted data (CED) [94], a sender sends data to a receiver to operate but does not want the receiver to learn anything about the data. In return, the receiver does not want the sender to know anything about the operating function. *Fully Homomorphic Encryption* (FHE) [95] is a solution for CED. In Figure 16, the sender first encrypts the data and sends them to the receiver. In return, the receiver operates the encrypted data and provides an encrypted output back to the sender to decrypt the result. The result of the

decryption is the same result as if the receiver would operate on the plaintext data.

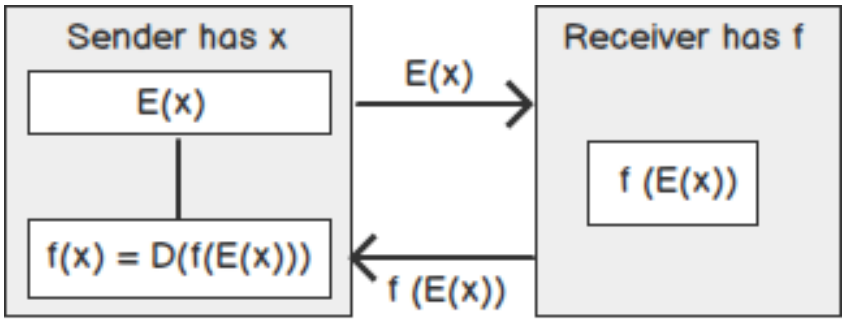

Figure 16: Homomorphic Encryption

Discussion: Although recent work in [96] improves the performance of FHE, it is still not practical for use (e.g., two multiplications encrypting a single bit takes 3.7s). Even if the performance of FHE were acceptable, the research would not investigate using FHE due to the following reason. The PDP evaluates the encrypted PII, but it has to send the result back to the data owner to decrypt the authorisation result. One may think to give the decryption key to the PEP so that the PEP can decrypt the result by itself. However, it is not acceptable because, in the concept of CED, the data owner should not disclose the description key to the receiver (i.e., PDP and PEP) [94].

4.4.2 Garbled Circuits

In secure computing with encrypted data and encrypted function (CEDEF) [94], a sender and a receiver wants to know the result of a joint function but does not want the other party to learn anything about its input. The sender may not want the receiver to know about the function. *Garbled Circuit* is one solution for CEDEF. Figure 17 illustrates the Garbled Circuits protocol [97]:

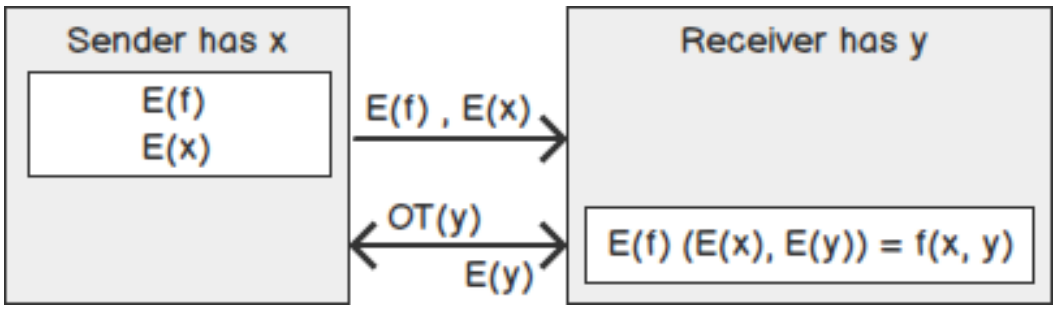

Figure 17: Garbled circuits

A sender can express an arbitrary function f that it wants to execute securely on a receiver in Boolean circuits. The sender then encrypts the circuits by mapping all Boolean values with some random strings resulting in an encrypted function *E(f)*. In particular, it maps all input wires and intermediate output gates, except the circuit output gates. The sender then keeps the mapping table secret and sends the Garbled Circuits *E(f)* to the receiver. The receiver interacts with the sender (via the *Oblivious Transfer* protocol) to ask for its garbled inputs *E(y)* without letting the receiver know the real inputs *y*. Now the receiver has *E(f)* as well as the garbled inputs of the sender *E(x)* and the receiver *E(y)*. With two inputs from both sides, it can open each gate of the circuits to compute the output gates. Since the receiver does not know the mapping table, it can observe the garbled output of each intermediate gate but cannot learn the real Boolean value. Because the output gates were not encrypted, the receiver can learn the output result. Recently the protocol has significant improvements in the circuit construction to evaluate billion circuit gates [98]. However, it still requires sufficient computing power (e.g., the edit distance between two 4095-bit strings requires a circuit of over 5.9 billion gates and 8.2 hours [98]).

Discussion: One can use Garbled Circuits in the following way. The PIP is the sender and the PDP is the receiver. The PDP can evaluate the garbled inputs of the PIP and returns plaintext output to the PEP without learning about the inputs. Although one may use Garbled Circuits to execute an arbitrary function

securely on the PDP, the research does not use it due to the following reasons. In FIDM, the PIP and the PDP may belong to various IDM systems. In FIDM, the PIP from one IDM system has to create Garbled Circuits for every PDP (of an SP) in another domain. Afterwards, the PDP has to interact with the PIP via the Oblivious Transfer protocol. However, it is not practical because the construction of the gates for each SP could be huge (i.e., not scalable) and the transfer protocol over the network may take a long time or break up in the middle. Indeed, work in [99] proposed to shift the creation of the Garbled Circuits from the sender to the receiver and embedded it in a shield location of the receiver by using trusted computing. This approach removes the interaction between the sender and the receiver but computes arbitrary functions securely and locally. However, the research looks for an approach that is independent of any hardware requirements.

In this section, the research discussed two interesting methods of secure computing. Although one may use Garbled Circuits, it is not scalable and requires intensive network interactions between the PIP and the PDP during the computation. In the next section, the research reviews a relatively new area in encryption to protect PII on an untrusted host.

4.5 Functional Encryption

Recently, cloud storage offers service for users to store personal data on the cloud (e.g., Amazon S3). One way to protect data on the cloud is to encrypt the data and allow those who have the secret key to decrypt them. However, this solution has the following limitations [100]: Whenever users want to process their data, they have to download them first, then decrypt, process, and re-encrypt them. Alternatively, users may send their secret keys to the server to

decrypt the data. The first case is inefficient for big data, and the latter discloses the data to the server.

Functional Encryption (FE) is a modern approach to solve the above issue. Figure 18 shows an overview of FE. Data owners encrypt their data (*m)* on the server, denoted by *E(m)*. When a user wants to perform a function *f* on the data, he submits a secret key sk_f that is associated with the function *f* to the server and gets the plaintext output of the function *f(m)* in return [101]. In this case, the server learns nothing about the data *m* but only the result of *f(m)*.

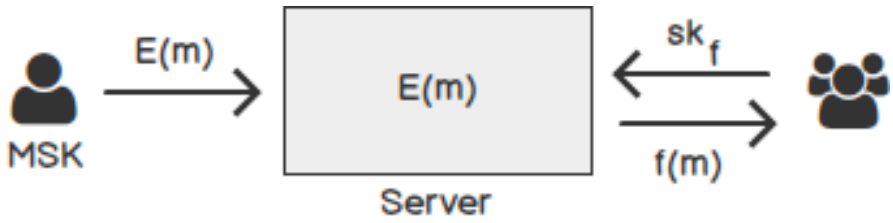

Figure 18: Functional Encryption

Predicate Encryption (PE) [101] is a special case of FE, whereby the encrypted data *E(m)* is associated with an *index*. If a user wants to decrypt the data, he will obtain a secret key sk_p that is associated with a predicate *p* and submits sk_p to the server. If the function p(index) evaluates to true, the server can decrypt E(m). PE has two subclasses: PE with public index and private index. In public index, the index is visible to everyone, including the server. In private index, the server can learn about the result of the function p(index) but nothing else about the index.

4.5.1 Attribute-based Encryption

PE with public index was first introduced in *Key-Policy Attribute-based Encryption* (KP-ABE) [100]. In KP-ABE, the index has a set of attributes, and the secret key sk_p holds an access policy. The server can decrypt the ciphertext if the attributes satisfy the access policy. Figure 19 illustrates an example of ABE.

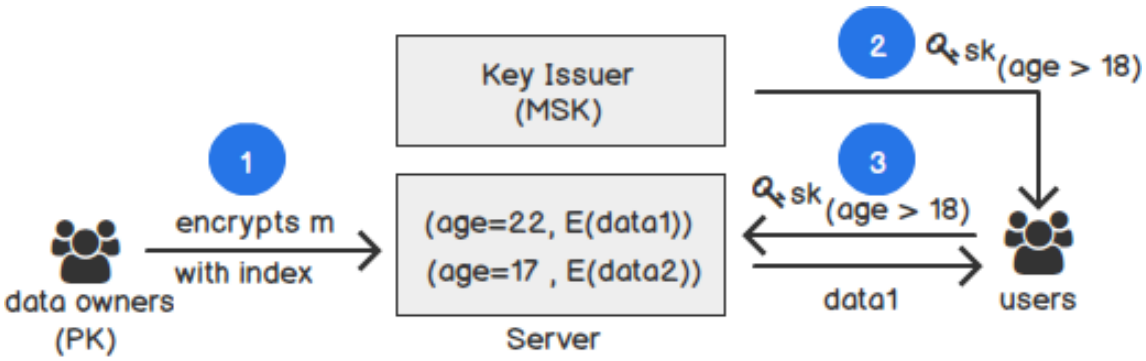

Figure 19: An example of Attribute-based encryption

First, data owners encrypt their data and store it on the server (e.g., the first and second ciphertext have an index `age` of 22 and 17 years old, respectively). Second, a user retrieves a secret key with an access policy `age>18` from a Key Issuer and submits it to the server. With the given key, the server can decrypt the first ciphertext `(age=22)`, which satisfies the access policy. In general, the ABE scheme has the following procedures:

1. **Setup** — The setup outputs a public key *PK* and a master secret key *MSK*.

2. **Encryption** — *CT ← Enc (PK, M, S)* — A data owner uses the public key *PK* to encrypt his data *m* with a set of attributes *S*.

3. **Key generation** — $sk_p \leftarrow Gen\ (MSK, P)$ — A key issuer uses the master secret key *MSK* to generate a secret key sk_p with an access policy *P*.

4. **Decryption** — $m \leftarrow Dec\ (sk_p, CT)$ — A user uses the secret key sk_p to decrypt the ciphertext if the policy *P(S)* evaluates to true.

Access policy

The access policy *P* can be expressed by a conjunction or a disjunction of equality test or inequality test (e.g., role = manager AND salary > 5000) and allows only a monotone access structure (i.e., no negative condition) [100].

Without going into much detail, Figure 20[4] illustrates how a decryption key is constructed from an access policy based on secret sharing.

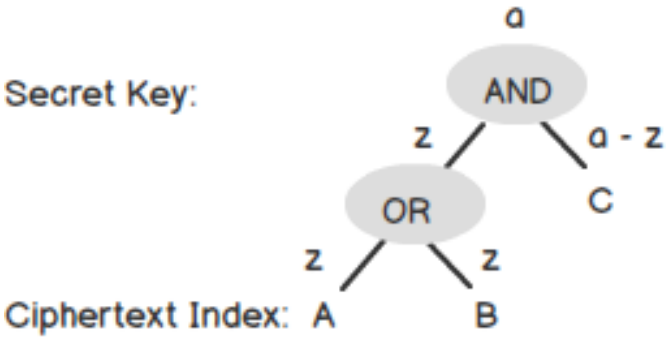

Figure 20: How ABE works without too much detail

In the key generation phase, a decryption key on the top α is split into different shares according to the access policy. For example, the decryption key α is split to z and α – z according to an AND operation, in which z is a random number. If ones have the secret keys with a set of attributes (e.g., A, C) that satisfy the access policy (e.g., (A OR B) AND C), then they have enough shares to reconstruct the decryption key α to decrypt the ciphertext. Here the random number z is in the exponent of the secret keys and will be cancelled (i.e., $\alpha - z + z = \alpha$). Most importantly, different users receive different secret keys, says sk_{p1} and sk_{p2}, with different random numbers. Because the random numbers of different secret keys are not the same, users cannot combine their secret keys to do the cancelling in the exponent. This will prevent different users to collude different secret keys sk_{p1} and sk_{p2} to generate a new decryption key that has more capabilities than what they are allowed them to have (e.g., $p_1 \cup p_2$ has more capability than p_1).

[4] Figure 20 illustrates the ABE scheme for an easy understanding. The figure does not show how the scheme works correctly. Please reference to [100] for details.

Key-Policy vs. Ciphertext-policy

There is another variation of ABE, the *Ciphertext-Policy ABE* (CP-ABE) [102], which is the reverse case of KP-ABE. In CP-ABE, data are encrypted with an access policy and the secret keys are associated with attributes. KP-ABE and CP-ABE target different applications. In KP-ABE, data owners have no prior-knowledge who will access their data. Thus, they can only encrypt their data with attributes relevant to the data and rely on a Key Issuer to decide later. In CP-ABE, data owners must determine who will access their data and encrypt them with an access policy beforehand.

4.5.2 Inner-product Predicate Encryption

PE with private index was first introduced in *Inner-product Predicate Encryption* (IPE) [103] and work continued in [104-107]. In IPE, data are encrypted with a vector $\vec{v}$ and a secret key holds a vector $\vec{w}$. The secret key can decrypt the ciphertext if the two vectors are orthogonal ($\vec{v} \times \vec{w} = 0$). In comparison to ABE, IPE can preserve the privacy of the attribute index, but the access policy of IPE is less expressive. In particular, IPE has to formulate the access policy in a *Conjunctive Normal Form* or a *Disjunctive Normal Form* [106]. For instance, if ones want to express a numerical comparison *(x<18)*, they have to convert it to *(x=1 OR x=2 OR… x=17)*. This causes a super-polynomial grow in size incurred by too many *OR* terms [106]. Work in [108] applied IPE to encrypt data and their experiment took over 20 seconds to generate a secret key for 8 attributes with 5 attribute values each. This performance is not acceptable for handling user authentication and authorisation.

The research also notes several techniques in ABE such as key revocation [109-111], a large universe of attributes [112, 113], and multi-authorities [110, 114, 115] as follows:

- In key revocation [109-111], a user encrypts his data with a *creation time t* and a receiver can decrypt the ciphertext if he has a secret key with a later time t' such that $t \leq t'$. The concept of IDaaS borrows this idea for defining an access policy but adds an *expiration time* to the ciphertext as well (i.e., *ciphertext creation time ≤ decryption time ≤ ciphertext expiration time*).
- In a large universe of attributes [112, 113], these schemes do not require setting up a fixed number of attributes (in the setup phase) that could be used to encrypt a message later. This is an important feature because the concept of using IDaaS needs to encrypt PII with multiple attributes (i.e., purposes, flexible domains, and time).

Finally, researchers found the ABE scheme of Rouselakis and Waters [115] that supports a large universe of attributes and multi-authority at the same time.

4.6 Conclusion

This chapter reviewed two categories for privacy-preserving user identity: the anonymity approaches and confidentiality approaches. While the first category satisfies the privacy properties undetectability and unlinkability, it does not support identity propagation in federated domains. They also overload the IDM tasks to the user's decision in every transaction.

In contrast to the anonymity approaches, the confidentiality approaches do not protect user anonymity but disclose PII to an entity if a disclosure policy is satisfied. In this category, most approaches cannot protect PII against an untrusted host. On the other hand, Mont *et al.* [81] draws the researchers'

attention as it can protect PII against untrusted host but has a limitation that only discloses PII to a target SP. In FIDM, intermediate entities in the call chain also need access PII as well. In comparison to the confidentiality approaches above, the concept of IDaaS borrows their ideas to encrypt PII with the disclosure policy to protect PII against untrusted host but enables the disclosure of PII over intermediaries.

Researchers also reviewed Purpose-based Access Control (PBAC). In comparison to the traditional RBAC, PBAC is suitable for a large distributed and heterogeneous environment. However, these solutions have difficulties in determining an access purpose of a request. The thesis may use PBAC for access control but may solve the limitation of determining an access purpose of a request. Moreover, related work aims at the local system only and is not concerned with federated domains. The thesis may propose a reference architecture for IDaaS, whereby PBAC also works in multiple federated domains.

Researchers also reviewed secure computation for protecting PII while computing a function on the ciphertext. Although one may use Garbled Circuits, it is not scalable and requires intensive network interactions between the PIP and the PDP during the computation.

In the last section, researchers reviewed Functional Encryption. Among several schemes of Functional Encryption, the Predicate Encryption with public index is faster and more expressive over the private one. The thesis may use this scheme to enhance the confidentiality approach of Mont *et al.* [81].

This chapter has detailed several approaches in many areas, whereby the combination of them may solve the research challenge of privacy-preserving

user identity. Chapter 7 will show the combination of sticky policy, PBAC, and CP-ABE to solve this challenge.

Chapter 5

Identity propagation

This chapter explains the identity propagation between service providers and reviews the design patterns for identity propagation: identity proxying (Section 5.1), impersonation (Section 5.2), forwarding (Section 5.3), and delegation (Section 5.4).

A *resource owner* (i.e., a user who owns a resource) has a set of rights (e.g., right to create, use, maintain, and billing). He may transfer a subset of rights to another entity to act on behalf of him. The following chapter reviews the *identity propagation* and explains the ownership transmission of a resource owner between services.

The following example explains the usage of identity propagation: A three-tier system consists of a front-end Web server, an application server, and a database in the backend. A user first authenticates to the front-end Web server by providing an attribute assertion about the user (e.g., username, group, age). An *identity-aware* application is an application that relies on the details of the user identity to adjust its behaviour, to authorise the user to perform various operations, and to audit user actions [116]. The front-end Web server may also transmit the user identity further to the underlying application server or even to the database.

Identity propagation, by definition, is "the replication of an authenticated user identity through multiple business systems and processes" [117]. The example above only mentioned one use case. The OASIS specification "Condition for delegation restriction" [118] defined four types of identity propagation (i.e., *identity proxying*, *impersonation*, *forwarding*, and *delegation*) as explained below.

5.1 Identity proxying

Identity proxying happens when an IdP acts as a gateway that issues an assertion to its relying party based on a given assertion. In Figure 21, a user Alice directly interacts with two IdPs and an SP sequentially from left to right. She authenticates to the first IdP to receive an assertion (step 1) and submits it to the second one (step 2). The second IdP acts as a proxy gateway that transforms the received assertion for Alice to access the SP (step 3). In this case, the SP only trusts the proxy gateway for issuing assertions.

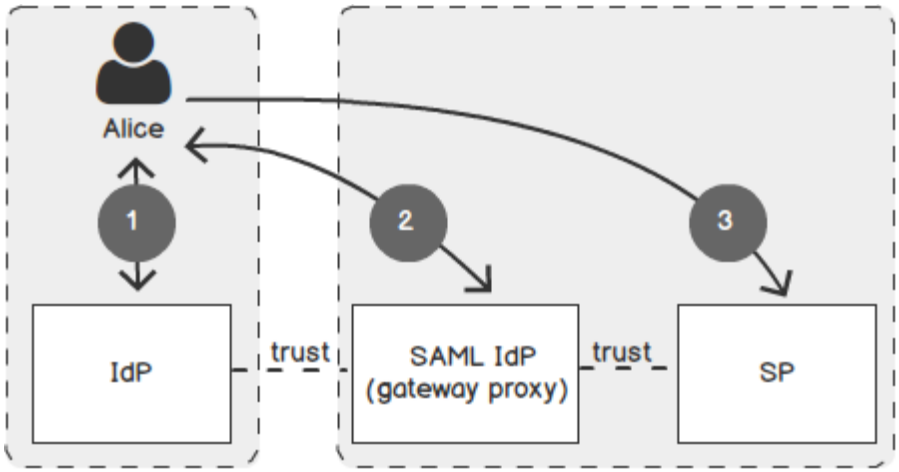

Figure 21: Identity Proxying

Trust model: Identity proxying is typically used when the user and the SP are in different security domains [119]. The user and the SP only trust its own IdP in the same security domain for issuing assertion, respectively. In addition, the IdP in one domain trusts the SAML assertion issued by the other one. A user can access a service in a different security domain because there is a trust relationship between the IdPs.

For identity proxying to work, the proxy gateway may support claim transformation to map user attributes from a *dialect* in one source domain to a dialect in a target domain that the relying SP can understand. For example, the proxying may map an element (e.g., `givenName`) defined in the standard schema

"System for Cross-Domain Identity Management" (SCIM) [120] to an element (e.g., `firstName`) that the SP can understand. Nowadays, most IdPs, such as WSO2 and OpenAM, support configurations for claim mapping [121].

5.2 Identity impersonation

An entity *impersonates* a subject or acts on behalf of the subject when it obtains a credential from the subject (e.g., username, password), or can get an assertion indistinguishable from an assertion that would be issued directly to the subject [118]. The entity has a complete set of permissions as the subject [122].

In Figure 22, a user Alice submits her credential (e.g., username and password) to a portal (step 1). The portal uses the credential to authenticate to an IdP (step 2) and receives a token in return to access a service backend on behalf of the user (step 3).

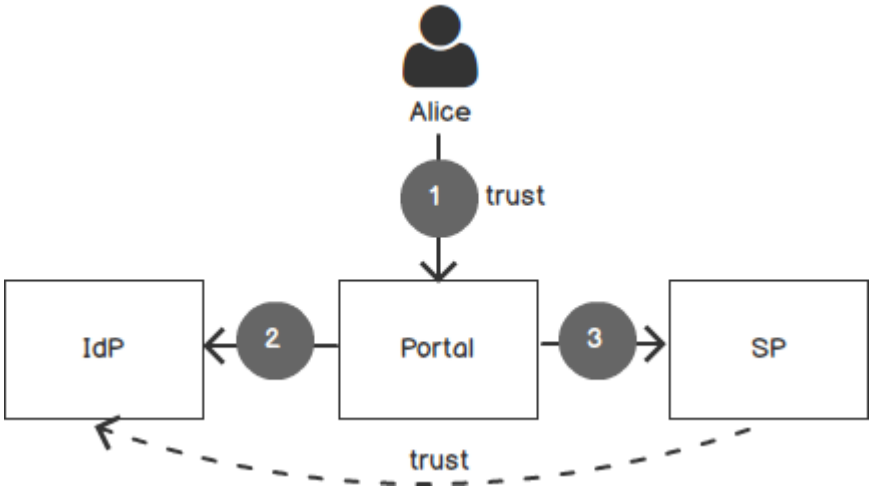

Figure 22: Impersonation

Because the portal obtains the user credential, it has the same permissions as the user to access the service backend. Therefore, ownership of a resource is transmitted to the portal completely. In an implementation, an intermediary may use the WS-Trust protocol [60] to request a token from the IdP (step 2). In particular, the intermediary embeds the user credential in the `OnBehalfOf`

element of the security token request. The IdP returns a new token that identifies user Alice in the `subject` element and contains no information about the intermediary.

Trust model: User Alice trusts the intermediate entity completely, and the SP trusts the IdP for issuing an authentication assertion.

5.3 Identity forwarding

Identity forwarding is a form of impersonation in which an intermediary entity receives an assertion from the subject and reuses it to impersonate a subject in the following call chain. The intermediary simply forwards the assertion without any modifications. Unlike identity impersonation, the intermediary cannot obtain the assertion by itself.

In Figure 23, a user Alice acquires an assertion from an IdP and sends to an SP1 for authentication, in step 1 and 2 respectively.

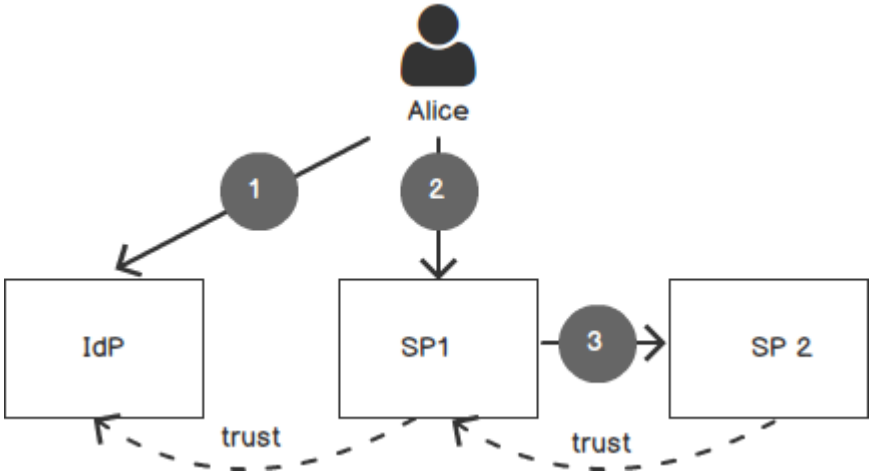

Figure 23: Identity Forwarding

In this scenario, the IdP issues an assertion only for SP1. When SP1 further calls SP2 (step 3), it passes the assertion further without any modifications. However, SP2 can verify that the assertion is issued for SP1, but not for the SP2 itself (e.g., by checking the *Audience Restriction* element in SAML [79]). If SP2

trusts service SP1, it may accept the requests from SP1. Otherwise, it will reject them.

Trust model: SP1 trusts its IdP for issuing an authentication assertion, and SP2 trusts SP1 for forwarding the authenticated user identity.

5.3.1 Identity forwarding in Java EE

Figure 24 illustrates a frequent use case in the Java EE [123]. An application client is making a call to an *Enterprise Java Bean* (EJB) in one application container, which in turn calls a second EJB in another container.

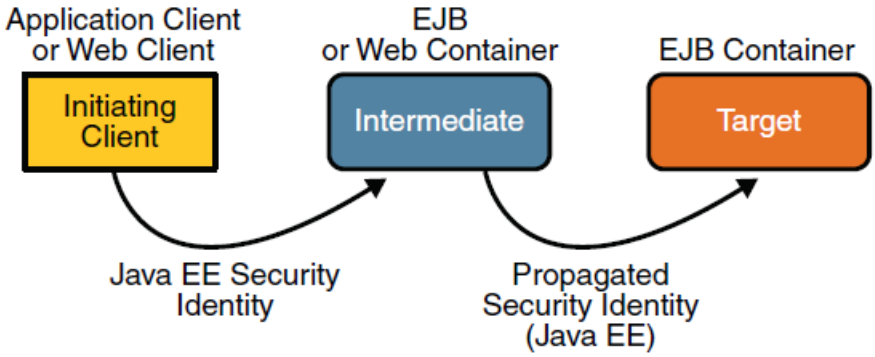

Figure 24: Security Identity Propagation between EJBs

When a user authenticates successfully into the first container, a `Subject` (that represents the user) is associated with a `Principal` that contains the name of the current authenticated user. The `Subject` is also associated with one or more user roles. When the first EJB calls the second one, the first container forwards the `Subject` to the second one by default [123]. It means the first EJB does not call the second one with its own service's identity but with the identity of the original user. As a result, both containers use the identity of the original user for authorisation and auditing purpose.

The Java EE also provides developers with an API (i.e., the Java Authentication and Authorisation Service) to retrieve the user identity programmatically. Listing 1 shows an example of how to retrieve a caller identity within an EJB [123].

```
# The following bean has a method to get the caller name
@Stateless public class EmployeeServiceBean implements EmployeeService {
  # First it references to the session context of EJB
  @Resource SessionContext ctx;
  public void changePhoneNumber(...) {
  ...
  # Then it obtains the caller principal's name from the session context
  callerKey = ctx.getCallerPrincipal().getName();
```

Listing 1: An example of using the JAAS API

For this to work, the second container has to trust the first one. Trust between two containers (i.e., between a client and a server) can be established in a way that the trust store of the client trusts the certificate of the server and the trust store of the server trusts the certificate of the client as in [124].

In some cases, a client may not want to forward the user identity to the server. The container deployment descriptor has the configuration to control identity forwarding (i.e., the `sas-context` element [125]) with three options: `None` (not support identity forwarding), `Supported` (accept identity forwarding from intermediate one), and `Required` (require all beans to forward identity) [125].

5.3.2 Identity forwarding in .NET Framework

As part of the .NET Framework, the Windows Communication Foundation (WCF) is a framework for building service-oriented applications. Just like in Java EE, WCF also defines various trust levels for impersonation between a client and a server (the `TokenImpersonationLevel` element [126]). In comparison to Java EE, WCF distinguishes two trust levels for identity forwarding:

- `Identification`: a service can obtain the identity of the calling client, but it has to execute with its own service's identity. The service is not allowed to impersonate the initiating client any further. This is the default value in WCF.

- `Impersonation`: a service receives the identity of the initiating client and executes with the identity of the initiating client. Recall that this is the default value in EJB as explained above.

5.3.3 Identity forwarding in SOA

The Java EE and .NET Framework support identity forwarding between application containers. Developers do not need to write custom codes in the application logic to transfer user identity from one service to another one[5]. However, calls between heterogeneous services in Service-oriented Architecture (SOA) are different.

In SOA, identity forwarding can be done either by passing user identity as an argument in a Web service's method or in the SOAP header [122]. For the first case, if user identity is a required behaviour of the business logic, there is nothing wrong to provide parameters in a Web service call for passing user identity [126]. In this case, the user identity is passed in the body of the SOAP message. For the latter case, the user identity can be embedded in the SOAP header as a custom XML element, or a standard *WS-Security Username token* (without a user password) [122]. Listing 2 shows an example of how to use a username token to forward a user identity.

```
# Username token inside a SOAP header to identify a username Alice
<env:Header>
  <wsse:Security>
    <wsse:UsernameToken>
       <wsse:Username>Alice</wsse:Username>
    </wsse:UsernameToken>
  </wsse:Security>
</env:Header>
```

Listing 2: Identity propagation using WS-Security Username token

[5] In Java EE, the containers forward user identity automatically only when developers use the container-managed authentication.

Whether a client service forwards a user identity in the SOAP header or the body, it must sign the message with its private key. By verifying the signature of the message, the partner service trusts the client service for generating a self-signed identity token [127].

If a target service does not trust the client service for a self-signed identity token, the client service may forward a SAML assertion, which was issued previously by a trusted IdP [122]. In Listing 3, a client service embeds a SAML assertion in the SOAP header of the message when it invokes a function `getQuote` of a partner service. The SAML assertion may contain information about the user such as email address to identify the user (in the `Subject` element) and additional user attributes (in the `AttributeStatement` element).

```
# Forward a signed SAML assertion in the SOAP header
<env:Header>
  <wsse:Security env:mustUnderstand="1" ...>
    <saml:Assertion AssertionID="Assertion-uuidf704f951-0104-f735-d1bb...">
      ...
      <saml:Conditions NotBefore="2005-07-08T15:15:35Z"
                       NotOnOrAfter="2005-07-08T15:26:35Z">
        # Identity Provider issued this assertion for the client service
        <saml:AudienceRestrictionCondition>
          <saml:Audience>http://clientService.com</saml:Audience>
        </saml:AudienceRestrictionCondition>
      </saml:Conditions>
      <saml:AttributeStatement>
        # An email address to identify user alice
        <saml:Subject>
          <saml:NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress">alice@service1.com</saml:NameIdentifier>
        </saml:Subject>
        # And further user's attributes required by the partner service
        <saml:Attribute AttributeName="...">
          <saml:AttributeValue>...</saml:AttributeValue>
        </saml:Attribute>
      </saml:AttributeStatement>
      <ds:signature>...</ds:signature>
    </saml:Assertion>
  </wsse:Security>
</env:Header>
<env:Body>
  <s:getQuote xmlns:s="http://PartnerService">
  ...
  </s:getQuote>
</env:Body>
```

Listing 3: Identity forwarding using SAML

In the above example, the IdP authenticates the user `alice@service1.com` and issues the SAML assertion for the intended client service `http://clientService.com` (as specified by the element `Audience`). The client service simply forwards the SAML assertion (without modifications) to the partner service, but the `audience restriction` remains as the client service.

In all cases, the client has to extract the user identity from the security context of the container and embed it in a SOAP request message. Upon receiving the SOAP request, the partner service has to parse the message to retrieve the user identity. Therefore, the partner service has to understand the format of the user identity delivered in the message beforehand. Both the client and the partner service usually negotiate the format of the message in their service contracts at the beginning [2].

For example, developers of the Java EE applications may implement a custom JAAS `LoginModule` to verify the SAML assertion for authenticity. This `LoginModule` may perform a mapping from the user identity delivered in the SOAP message to its security context of the container. Because the Java EE does not support a SAML `Principle`, developers may implement a custom `Principle` (for the corresponding SAML assertion) and associate the `Subject` with the `Principal` (by using a `commit()` method in JAAS) [128].

5.4 Identity delegation

In the forwarding scenario, an intermediate entity forwards an assertion down to its partner service without modifications. Identity delegation extends the forwarding scenario by adding information to the assertion that identifies all parties (i.e., the user, all intermediate entities, and the partner service) in a transaction. To identify all parties in a transaction, either an IdP identifies all

parties and issues a complete assertion at the beginning, or intermediate entities in each hop send a request back to the IdP asking for a *delegate assertion* to access the partner service [118].

In Figure 25, SP1 calls SP2 on behalf of a user Alice.

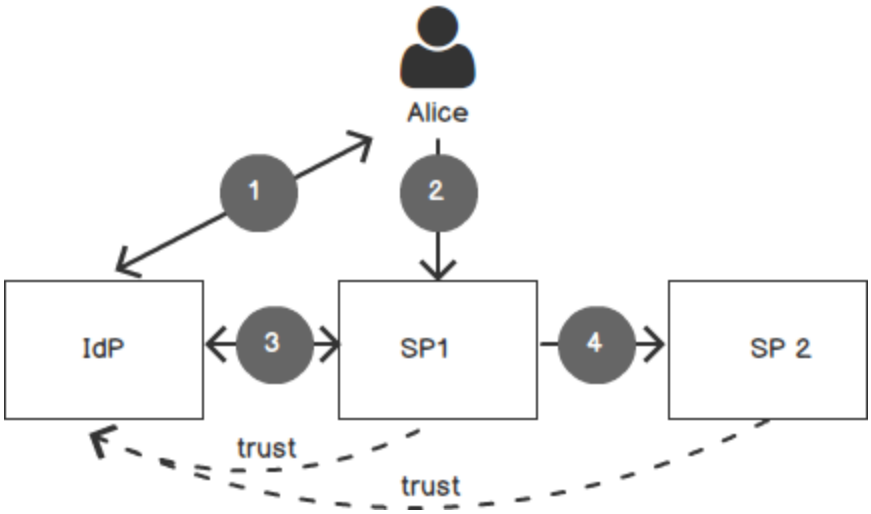

Figure 25: Identity delegation

In the beginning, Alice requests an authentication assertion from an IdP (step 1) and accesses SP1 with this authentication assertion (step 2). SP1 resubmits the authentication assertion received from Alice to request a *delegate assertion* for accessing SP2 (step 3). The IdP already has Alice's policy to allow SP1 to access a resource on SP2 on behalf of Alice, so it responses SP1 with a delegate assertion. SP1 finally accesses SP2 with the delegate assertion (step 4). SP1 typically uses the certificate of SP2 to encrypt the request message for protecting the delegate assertion for confidentiality.

Trust model: both SP1 and SP2 trust their IdP. SP2 does not trust SP1 and must validate the delegate assertion issued by the IdP.

5.4.1 Identity delegation with WS-Trust

In SOA, an SP may use the WS-Trust protocol (version 1.4) [60], which has an `ActAs` element for expressing a delegate assertion. In particular, SP1 sends a

security token request to the IdP with an `ActAs` element containing the whole SAML response that it received from the user Alice (step 3). The IdP then responses with a delegate assertion, which identifies the service SP1 in the `Subject`, user Alice in the `ActAs` element, and the role of the user Alice (i.e., the IdP states that SP1 is allowed to act on behalf of user Alice for a given role) [129]. In comparison to the `OnBehalfOf` element in *Identity impersonation*, the `ActAs` element identifies both the intermediary SP1 as well as the user Alice in the returned token.

5.4.2 Identity delegation with OAuth 2.0

Alternatively, one may use the protocol Open Authorization (OAuth) 2.0 [72] for identity delegation with the same scenario: SP1 needs to access a resource in SP2 on behalf of Alice. Alice may grant an *access token* from SP2 for SP1 by following the workflow "Authorisation code grant type" in OAuth 2.0 [72]. In this workflow, SP1 stores the access token (and a refresh token) corresponding to the username Alice locally so that it can act on behalf of Alice in future calls to access SP2. This solution has an overhead for SP1 to store an access token for each user [119].

Alternatively, the "SAML 2.0 Profile for OAuth 2.0 Client Authentication and Authorisation Grants" [71] offers an alternative implementation. In this profile, an SP receives a SAML assertion and exchanges it for an access token to a downstream service in another domain (Figure 26).

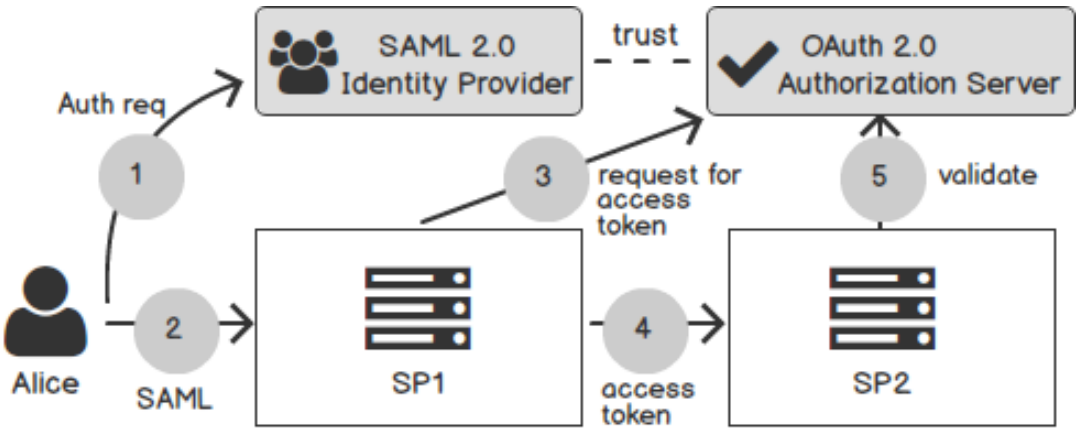

Figure 26: Exchange a SAML token for an OAuth access token

In Figure 26, SP1 accesses SP2 on behalf of user Alice. SP1 first receives a SAML assertion from user Alice (step 1 and 2). Then it exchanges the SAML for an OAuth access token by contacting the authorisation server of SP2 (step 3). The authorisation server validates the SAML and issues an OAuth access token. Finally, SP1 uses the access token for accessing SP2 (step 4). SP2 validates the access token with its authorisation server (step 5). In this design pattern, the authorisation server trusts the IdP for issuing the SAML. In comparison to the first solution, SP1 does not need to manage an access token (and a refresh token) for each user but requests an access token at runtime.

5.5 Conclusion

This chapter clarifies the needs for propagating PII over intermediaries in the call chain. An identity-aware application relies on PII to adjust its behaviour, to authorise users, or to audit users' actions.

This chapter explained the adaptation process for propagating PII between SPs in further details (e.g., in Java EE, in .NET framework, and in SOA). This time-consuming task could take place in several months. IDaaS removes these burdens from the application developers and facilitates the identity propagation between SPs on demand.

Chapter 6

Security design patterns

Chapter 6 reviews the security design patterns for authentication (Section 6.1) and authorisation (Section 6.2) and examines how to decouple an AAI implementation from an application (Section 6.3).

6.1 Authentication patterns

Authentication is the process of validating user-provided credentials to prove that users are who they claim to be [119]. There are two main authentication models: *direct authentication* and *broker authentication* [127]:

6.1.1 Direct authentication

In direct authentication, an initiating client accesses the SP by providing a credential (e.g., username and password) for authentication. Direct authentication affects the performance because the client authenticates to the SP in every request (i.e., no Single-Sign-On supported) [127].

6.1.2 Broker authentication

In broker authentication, the client and the SP trust a third party for an authentication request. The client authenticates to the third party and receives a token in return. Then it caches the token and uses it to access multiple SPs who trust the third party. The following parts briefly summarise some frequently used tokens with their key aspects.

Kerberos

In the Kerberos protocol [130], the client and the SPs share a secret key with a *Key Distribution Center* (KDC), which acts as a TTP. The KDC uses the secret key to distribute a session key and a service ticket (i.e., a session-based token) to the client and the SP in contact. Then the client and the SP can use the session key as a symmetric key for mutual authentication, for instance in 8

hours, until the service token expires. Kerberos does not support revoking a service ticket. Because the client shares the same secret key with the KDC, one cannot use the Kerberos protocol for proving the non-repudiation of the client. Furthermore, the Kerberos protocol is typically limited within one organisation [127].

X.509 PKI

In X.509 PKI [65], the client as well as the SP, each entity possesses an asymmetric key (i.e., a secret and a public key) issued by a trusted certificate authority (CA). The client uses its secret key to sign the request message, and the SP uses the client certificate (i.e., the public key of the client) to validate the signature of the message. If the signature is valid, the SP authenticates the client request. The client may attach its X.509 certificate in the request message if the SP does not have it. X.509 can be used for authentication at the transport layer (e.g., use HTTP over SSL [123]) or at the message layer (e.g., use Web service X.509 token profile [131]).

In comparison to Kerberos, one can use X.509 to prove for non-repudiation of the client, since only the client possesses the secret key. Furthermore, the SP does not require the CA to be available for issuing a ticket request like Kerberos [127].

By using the X.509 v3 certificates with attribute extensions, the CA signs the user's public key together with the user attributes and includes the signature in the certificate. Then the SP may use the user attributes in the X.509 certificate for access control. Because the X.509 certificate is usually valid for a long time, it is not suitable for access control that changes frequently (e.g., in RBAC, a user role may frequently change when he moves to another department).

Furthermore, the user has to reveal unnecessary user attributes in the certificate, which fails to protect user privacy [34].

Security Token Service

The Cloud Security Alliance recommends using SAML as a standard protocol to transfer a user identity from one entity (e.g., an IdP) to another one (e.g., an SP) [28]. In comparison to X.509 PKI, a Security Token Service (STS) issues an assertion with the required user attributes only. The assertion also has a short lifetime and is suitable for frequent policy changes. Unlike Kerberos, SAML can be used across boundary domains [127].

6.2 Authorisation patterns

An authorisation is a process of validating what actions an authenticated user can perform in a system [119]. In a one-tier architecture, all required components of an application are in one server. User information and the security policies are on the same server for performing access control. The concept of multi-tier architecture physically separates an application into multiple tiers (i.e., presentation, application processing, and data management tier) so that developers can create flexible and reusable applications [132]. However, multi-tier architecture also introduces problems in the management of user identity and security policies in each tier as explained below [133]:

- The first issue is identity propagation. For example, a traditional Database Management System (DBMS) requires user identity in order to control access to user data in the database. In this case, one needs a mechanism to forward the user identity from the front-end through all middle tiers to the DBMS. In the previous chapter, a middle tier may impersonate the original user, open a connection, and authenticate to the DBMS. The DBMS has to

trust all middle tiers to forward the actual user identity, but not any other's identities. Furthermore, all middle tiers and the DBMS must share the same security context (i.e., they share the same semantic meaning of a user attribute).

- The second issue is performance. Each user establishes a connection and authenticates to the DBMS. However, establishing and maintaining a connection for each user to the DBMS is expensive.

To solve the problems above, work in [70, 71, 78] introduced two authorisation design patterns: a *Trusted Sub System* and a *Delegated Access Control* in the following way.

6.2.1 Trusted subsystem

By definition, a trusted subsystem model is a service that performs authorisation for a certain application task so that a downstream service, which trusts the subsystem, does not need to perform the required authorisation again [122]. The trusted subsystem requires the user identity for its decision. Because the downstream service does not perform authorisation again, it is not mandatory to propagate the user identity from the trusted subsystem further to the downstream service [116].

Benefits

The trusted subsystem model is beneficial to solve the performance problem as follows. In Figure 27, a database server trusts a Web service as its trusted subsystem. Firstly, the Web service checks if a user Alice can access her data in the database server. Secondly, the Web service does not need to impersonate Alice but authenticates to the DBMS with the Web service's identity.

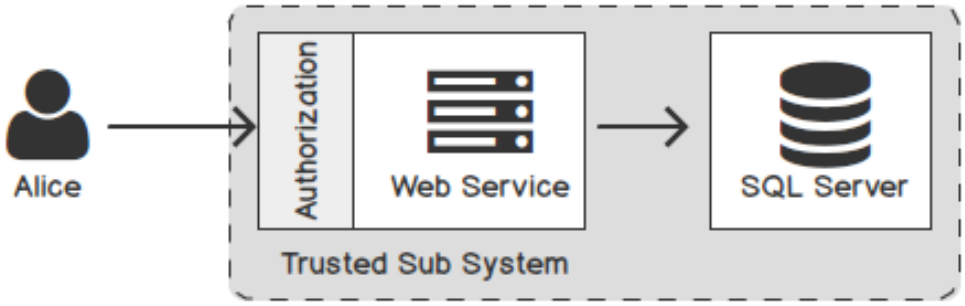

Figure 27: A trusted subsystem

As a result, the Web service can work on behalf of many users to query the information from the database by using a pool of connections. A pool of connections used by one service saves an expensive resource from opening, maintaining, and closing connection for each authenticated user [133]. On the other hand, this model reduces the complexity of the downstream service (e.g., for the DBMS) to manage access control for all authenticated users. Instead, the downstream service only manages access control for a few trusted subsystem services [122]. Finally, if all authenticated users have permissions to invoke a downstream service, it violates the principle of *defense in depth* [122]. According to this principle, the downstream service should restrict the callers to a small set of application identities only. Otherwise, any authenticated users may send requests to the DBMS, since they also have the credentials to do so.

Security requirements

The trusted subsystem model has to fulfil the following requirements [122]:

1. The trusted subsystem and its downstream service are mutually authenticated with each other.
2. The communication between the trusted subsystem and the downstream services must be protected from a man in the middle for confidentiality and integrity.
3. The downstream services have policies to control which functions a trusted subsystem can call.

One can fulfil the first two requirements by several approaches [127]: For examples, one may use SSL/TLS to protect the communication at layer 4 (i.e., the transport layer). Alternatively, one may use IPSec to protect the traffic between two machines at layer 3 (i.e., the network layer). IPSec has a better performance than message layer security, but the trust relationship happens between two machines, and not between two applications for fine-grained access control.

For the third requirement, one may control access to the downstream services on the application layer. For example, the Java EE offers a possibility for an EJB to call a target EJB with a specific `Principle` other than the initiating client. In Listing 4, developers can configure an EJB to call a target EJB with a different role (e.g., an `admin` role):

```
# calls to methods in this class will proceed with a new role admin
@RunAs("admin")
public class Calculator { ... }
```

Listing 4: Identity propagating with a different identity in EJBs

Then, the target EJB defines permissions to a set of functions based on the specific role (e.g., `admin`), which belongs only to its trusted subsystem. Because authenticated users do not have the `admin` role, they cannot bypass the first EJB and access the second EJB directly [123].

Limitation

However, there is a security risk for this model when the trusted subsystem is compromised [127]. An attacker may gain access to the trusted subsystem and use the credential of the trusted subsystem to further access resources of the downstream services legally. For example, an attacker can query all user data in the database from inside the hosting of the Web service, although there is a firewall between the Web service and the DBMS. Therefore, an API gateway

that acts as a trusted subsystem for authentication and authorisation is usually secured in a separated *Demilitarized Zone* (DMZ). Moreover, the downstream service is in located in a *Militarized Zone* (MZ) with different protection requirements [119].

Identity forwarding is optional

Recall that in the trusted subsystem, it is not mandatory to forward user identity to the downstream services. However, in many cases, the downstream services may require the user identity not only for auditing purpose but also for protection against replay attack as follows [127].

In *Identity forwarding*, the trusted subsystem may forward a SAML assertion issued by an IdP to the downstream service. The downstream service, which trusts the IdP, verifies the `condition` element if the SAML assertion is still valid (i.e., the downstream service does not re-authenticate the original caller again but check if the assertion is still valid). In case the trusted subsystem is compromised, the attacker may cache the assertion and replay it in a subsequent request. However, the downstream service can reject a request with an outdated assertion and thus limits the consequence of a compromised trusted subsystem [122]. The SAML security specification recommends to set the valid time of an assertion to a few minutes (e.g., 5 minutes) [62].

6.2.2 Delegated access control

Unlike the trusted subsystem, the downstream service does not rely on the identity of the upstream service to authenticate or to perform access control. Nevertheless, the downstream service relies on the user identity to re-authenticate the user and performs authorisation. Therefore, the upstream

service must propagate (i.e., either forward or delegate) the user identity to the downstream service [116].

Benefits

Figure 28 shows an example of delegated access control: SP1 is an intermediate service between user Alice and SP2. In this model, SP1 does not have the permission to access SP2 but propagates the identity of Alice (e.g., a SAML assertion) to SP2. If SP1 is compromised, an attacker cannot access SP2 immediately but waits for an incoming assertion from Alice to access SP2, which will slow down the attack [116].

Figure 28: Delegated access control

In the case of using identity delegation, Alice can control which intermediate services transmit her identity. For example, by using the OAuth protocol, Alice can grant permission for SP1 to act on her behalf to access a resource in SP2. In the trusted subsystem model, Alice is not aware which services are in the call chain because the upstream service simply forwards her identity to the downstream service with or without her permissions.

In best practices, it is recommended to use the trusted subsystem for low-privilege, high-volume transactions (e.g., normal users daily logins to a Web application), and to use delegated access control for high-privilege, low-volume transactions (e.g., admin operates system sometimes in a week) [116]. In general, the further down the call chain from the original user, the less relevant the user identity is [126].

6.3 Decouple Authentication and Authorisation

The following sections focus on existing strategies that decouple the security implementation from the application logic. The first section reviews an *intercepting Web agent* that protects a Web application running on the same Web server. The second section describes a *security gateway* that runs on a separated hosting and protects several SPs in the backend.

6.3.1 Intercepting Web agent

The core security patterns [128] presented an intercepting Web agent as a design strategy to protect Web applications that have little or no security. Developers often design such applications with a focus on functionality at the beginning and overlook or postpone security in the end. After the implementation completed and administrators deployed the application in an environment, it is hard to implement additional security mechanisms. So rather rewriting the Web application, administrators can deploy a Web agent running on the same Web server as the Web application. The Web agent intercepts a request to and response from the application to enforce authentication, authorisation, and auditing [128].

This design has the benefit to change security implementation outside the Web application without affecting the running application. In additions, it also increases the application performance, because the Web server now performs the security-related tasks [128].

The following parts explain the request flow of a Web agent and review its implementation in scripting languages, in Java EE, and in .NET Frameworks.

6.3.1.1 Request flows

Figure 29 shows the request flows, how a Web agent interacts with an IdP for user authentication and authorisation [128] [134]:

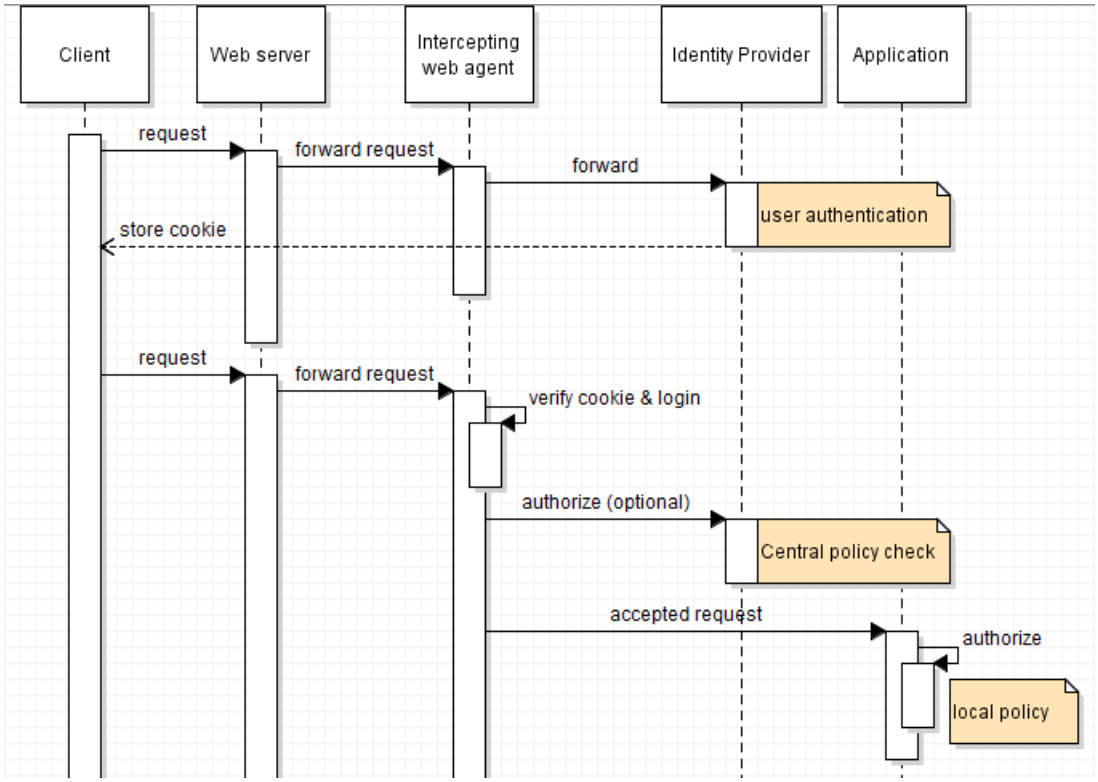

Figure 29: Intercepting Web agent request flows

1. A client sends a request to a Web application running on a Web server. If the client accesses a protected resource URI, the Web server forwards the client request to a Web agent.
2. The Web agent checks if the client request is authenticated (e.g., if a session cookie is present or not). If not, it forwards the client to an IdP.
3. The client authenticates to the IdP (e.g., with a username and password, not shown in the figure). If successful, the IdP may store a cookie with an encrypted session on the client browser (or issues a SAML assertion alternatively) and redirects the client back to the Web server.
4. The Web agent intercepts the client request on the Web server and checks if the session cookie (or the SAML assertion) is valid. If yes, the agent may authorise the request with the policies stored in the IdP. In this case, the

agent acts as a PEP and the IdP acts as a PDP according to the XACML reference architecture [6].

5. If the IdP accepts the authorisation request, the Web agent forwards the client request to the application. It may include in the client request with privileged information about the client (e.g., username, roles, and groups), and additional user's profile attributes (e.g., age, hair colour) that it received from the IdP for the application to use.
6. The application may perform additional authorisation based on the privileged information forwarded by the Web agent (e.g., RBAC).

In step 5 above, the Web agent may propagate the user identity (i.e., privileged attributes, or profile attributes) to the application in the backend. The implementation of identity propagation may depend on programming languages and platforms as follows:

6.3.1.2 Implementation in scripting languages

The Web agent may propagate the user identity by inserting custom HTTP headers to the client request before forwarding it to the application. The inserted HTTP headers vary from vendor to vendor. For instance, Cafesoft [135] inserts custom headers that prefixed by `CAM-` (e.g., `CAM-HTTP-USER`, `CAM-HTTP-ROLES`). For this to work, the Web agent has to treat all requests from the client with the same prefix headers as malicious requests [136]. In the backend, the application simply queries the HTTP headers for user identity as in Listing 5:

```
# a JSP application checks for client authentication by querying
# CAM-HTTP-SESSION-ID from HTTP headers [135]
if (request.getHeader("CAM-HTTP-SESSION-ID") != null) {
  ...
}
```

Listing 5: Querying user identity from HTTP headers

Identity propagation in HTTP headers is considered as programming and platform-independent for any applications to obtain information about a request

[136]. Therefore, any scripting languages (e.g., PERL, PHP, or Ruby) may use this method [136].

6.3.1.3 Implementation in Java EE

In Java EE, identity propagation is more convenient, because the application container supports it. Recall in *Identity forwarding in Java EE*, the containers forward the user `Subject` between applications in the call chain by default. Therefore, the Web agent only needs to commit the `Subject` into the container for management. The following parts explain the implementation in further details.

Servlet filter

Developers may use a Servlet Filter to implement a Web agent. Listing 6 shows a Servlet Filter that intercepts an HTTP request before forwarding the request further to the application in the backend [123].

```
# Servlet Filter calls doFilter() to handle request and response
public void doFilter(ServletRequest request, ServletResponse response,
                     FilterChain chain) {
    # Forward request to the application backend
    chain.doFilter(request, response);
    # After application returns, the Filter continues from here
    ...
}
```

Listing 6: Implementing an intercepting Web agent with Servlet Filter

The container-managed authentication

The JAAS specification describes the standard interfaces to authenticate a user into the container [137] [138]. In JAAS, a `Subject` represents a group of related information about an entity (e.g., a user) [138]. In Figure 30, a `Subject` has multiple identities (i.e., a `Principal`) such as username, login-id, and user groups to which a user belongs. A `Subject` also has public credentials to be shared (e.g., public key credentials or Kerberos server tickets) and private

credentials (e.g., private keys). Public and private credentials require different permissions to access and to modify [138].

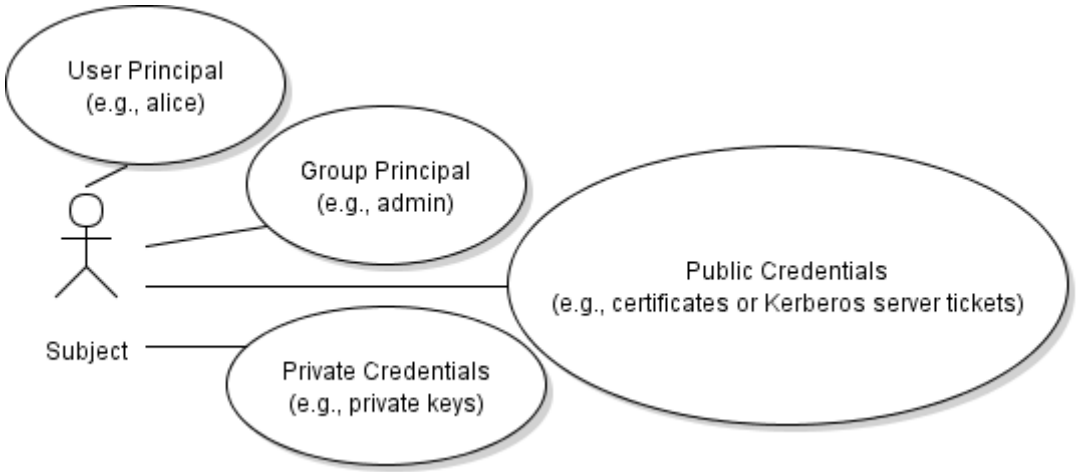

Figure 30: Subject and associated principal

In step 3 of the *Request flows*, the user submits a SAML assertion issued by the IdP to access the application. To authenticate the user request, the Web agent may implement a JAAS `LoginModule` that validates the SAML assertion and extracts the user `Principals` from the assertion. If the SAML assertion is valid, the Web agent implements the `commit()` method that associates the user `Principals` and the credentials with the `Subject` [128]. When the user logouts, the Web agent also implements a `logout()` method that removes the `Principals` and the credentials from the `Subject` [128].

Once the `Subject` is committed into the application container, the container is responsible for propagating the `Subject` and its `Principals` further down the call chain. With this, the application can query the user's privileged attributes via the JAAS APIs when needed (e.g., by using `getUserPrincipal()` method) [123]. It means developers do not need to implement an additional function to pass the caller's `Principals` between applications.

In the backend, the Web application (or EJBs) may define a list of mapping roles to the current caller's `Principals` in its deployment descriptor. In Listing 7, a

caller's `Principal` with the group name `manager` is mapped to the local role `manager_role`.

```
# map group 'manager' to role 'manager_role'.
<security-role-mapping>
  <role-name>manager_role</role-name>
  <group-name>manager</group-name>
</security-role-mapping>
```

Listing 7: An example of role mapping

Based on this role, the application may perform an RBAC. For example, it may check whether the caller has a specific `manager_role` (by using `isUserInRole()` method [124]) before it allows the caller to access a resource.

Portability of Java application

The following part reviews the portability issues of a Java EE application. The JAAS specification provides standard interfaces for container authentication, but the implementation of the JAAS `LoginModules` may use container-specific APIs and configurations that lock the application to a specific vendor [139]. This vendor lock-in limits the application's portability between containers from different vendors.

In addition, the JAAS specification provides standard `Principals` (as in Figure 30) but does not support for a `SAML Principal` (i.e., a `Principal` that holds user's profile attributes delivered from a SAML assertion). To support a `SAML Principal`, many containers provide their `Principal` implementations, which again lock an application to a specific container implementation [140]. Alternatively, an application server like JBoss supports a security context for an application to set a SAML assertion in it (i.e., `SecurityClient` [141]). Then the SAML assertion will be propagated together with the security context between JBoss application servers automatically [141]. Existing Web agents (e.g.,

OpenAM or Cafesoft) only supports forwarding user attributes from the agent to the application via HTTP headers or session cookies [134] [135].

Portability of the Web agent

The following part investigates whether the Web agent and the application can be migrated together to another cloud provider as a whole. The Web agent implementation often binds to the IdP of the same vendor in the following ways:

1. In step 2 of the *Request flows*: after an IdP authenticates a user, the Web agent may set session cookies in the user's browser (instead of issuing a SAML authentication assertion). Because cookies are not standardised, the IdP and the Web agent in such case must be compatible. In practice, they are often from the same vendor [23]. For example, the IdP from OpenAM [31] or Cafesoft [28] uses cookies, and thus these IdPs only works with their Web agent implementations.
2. In step 4, the authorisation request from the Web agent to the IdP (which acts as a PDP) is also not standardised. For example, the WSO2 IdP provides its interfaces for authorisation request using the *Thrift* protocol or Web services [121], but the OpenAM IdP provides its own REST APIs [134].

6.3.1.4 Implementation in .NET Framework

The previous section showed several limitations for implementing portable applications in Java EE. On the other hand, the Microsoft .NET framework provides the *Windows Identity Foundation* (WIF) [142] that helps developers to build an identity-aware application easily. The following section reviews how WIF improves the issues above in Java EE.

Request flows

WIF has similar *Request flows* of an intercepting Web agent previously. It means WIF intercepts client request and redirects the client to an IdP if not authenticated. However, unlike the Web agent of OpenAM [134], WIF does not use session cookie but expects a SAML token issued by the IdP [143]. SAML token has an advantage against cookies because the SAML token is standardised and thus compatible with multiple IdP vendors (i.e., no lock-in vendors) [23]. Furthermore, WIF can transform a given SAML token into its standardised identity model (below). Developers can use this identity model and do not need to understand the underlying protocols in details.

Identity model for claims

In .NET Framework, the interfaces `IIdentity` and `IPrincipal` represent the caller of service (see Figure 31 [142]). They are similar to the `Subject` and the `Principal` in Java EE that hold the identity information of the caller. The `IIdentity` represents the username and the information, how the user was authenticated. The `IPrincipal` provides information on whether the caller has a specific role. All of this information is available to the developers via a local call to the `Thread.CurrentPrincipal` [126]. So far, there is nothing new in comparison to Java EE.

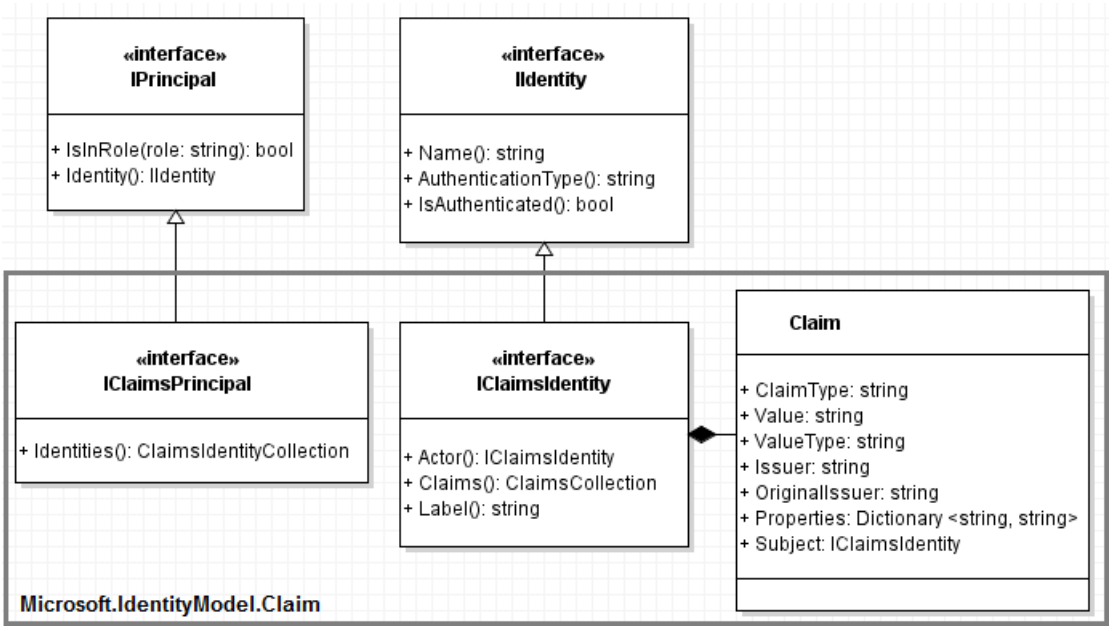

Figure 31: Identity model in WIF

WIF extends the existing identity model of .NET framework with two additional interfaces (i.e., `IClaimsPrincipal`, `IClaimsIdentity`) [143]. The interface `IClaimsPrincipal` exposes a collection of identities. Each of which in turn has a collection of claims about a caller. A `Claim` represents a single attribute statement about the caller asserted by an `Issuer` with the following attributes:

- `ClaimType` represents the type of claim in URI format. For instance, an e-mail address is represented by the type `http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`. It means WIF predefines all standard claims about the user attributes so that a claim of the same type has the same meaning in different security domains. This is what missing in the current Java `Principal`.
- `Properties` are dictionaries for developers to provide application-specific data about a user (e.g., the last ten items a user bought from a store). In practices, a central IdP should only store common attributes that are useful for multiple applications (e.g., privileged attributes) [144]. `Properties` are

useful for attributes that belong only to a specific application and nowhere else.

- In the interface `IClaimsIdentity`, the method `Actor()` returns the identity of an intermediate caller. This element implements the `ActAs` element of the WS-Trust protocol.

At a certain point when developers need to know the caller identity (e.g., developers need an email), they can query it as in Listing 8:

```
# Get caller's identity from the current Thread [143]
IClaimsIdentity identity = Thread.CurrentPrincipal.Identity as
                           IClaimsIdentity;
# From all claims in identity, get the claim value from type 'email'
string Email = (from c in identity.Claims
            where c.ClaimType == System.IdentityModel.Claims.ClaimTypes.Email
            select c.Value).SingleOrDefault();
```

Listing 8: Querying user identity in .NET framework

WIF suggests that developers define all required claims they need in the application's configuration so that a client authentication request can ask the IdP for all necessary claims [143]. However, identifying all required claims that an application needs to complete a request is not an easy task. Therefore, WIF supports developers with a wizard in Visual Studio tool [143].

6.3.2 Security gateway

The previous section reviewed an intercepting Web agent that protects a single Web application in the backend. When there are more than one services in the backend, the *core security patterns* [128] suggests using a *Secure Service Façade* that acts as a security gateway to protect multiple services for the following reasons:

- Having more access endpoints leads to more opportunities for security holes. Instead of implementing security protection for each service (i.e., authentication, authorisation and auditing), a central gateway enforces

general security requirements so that developers do not miss any requirements or security implementation updates in any services.

- Security developers and business developers can separately focus on the development of the gateway and business services, respectively.
- The gateway may offer a unified security API to the client and hide the complexity of each service's APIs in the backend.
- The gateway may minimise the message exchange between the client and the services by storing session states and security context on the gateway. The gateway may also orchestrate multiple services in the backend and aggregates the results before sending a response back to the client.

6.3.2.1 Request flows

Figure 32 shows the request flows of a security gateway [128] [145]. Briefly, it looks very similar to an intercepting Web agent: First, the client authenticates to the gateway (e.g., using a SAML assertion provided by an IdP). Second, the gateway may authorise the request by asking an IdP that acts as a central PDP (step 2). If the request is accepted, the gateway forwards it to an appropriate SP in the backend.

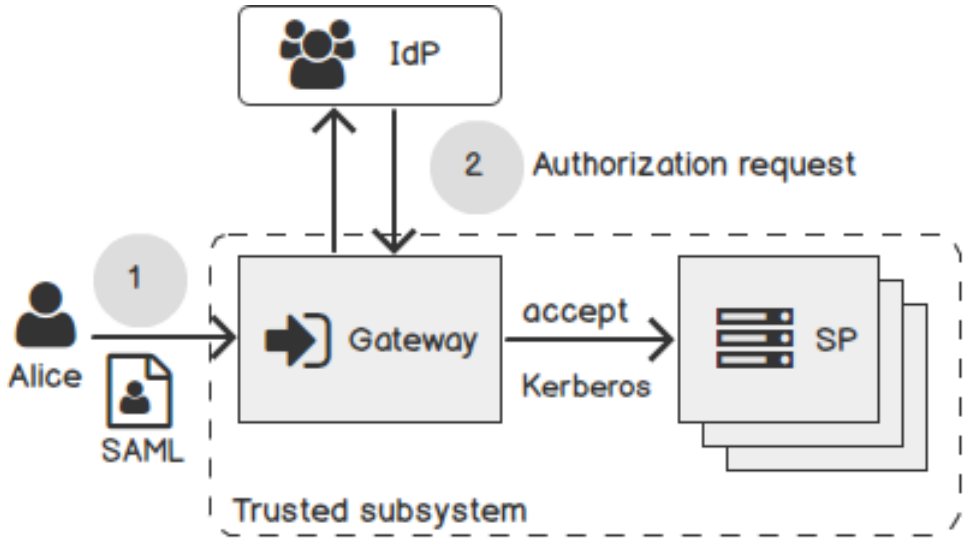

Figure 32: Security gateway protects multiple services in the backend

In general, the gateway acts as a *Trusted subsystem* that is responsible for enforcing authentication and authorisation for multiple services in the backend. Because the gateway is a trusted subsystem, it calls the SPs with its own service's identity (i.e., not with the identity of the original client). In the backend, the SPs authenticate the request coming from the gateway (i.e., SPs do not authenticate the request coming from the original client). In this case, the gateway bridges incompatibility of security protocols between the client and the service backend [145]. For example, the client may authenticate to the gateway with a SAML assertion, while an SP is implemented with WCF and supports Kerberos authentication. After the gateway validates the SAML assertion, it authenticates to the SP with Kerberos authentication.

6.3.2.2 Identity forwarding

Recall that in a trusted subsystem the gateway may forwards the identity of the original client to the SP if the SP requires it. In such case, it is important for the gateway to maintain a security context of the client (e.g., as a state variable in a stateful session bean) and propagate the required client identity to the SP when needed [128]. In case the gateway provides a workflow process that orchestrates multiple SPs in the backend, it is handy for a transaction management system to access the security attributes at the same place [128].

Recall that in *Identity forwarding*, an application container forwards user identity by default when the gateway and the SPs use the same implementation platform (e.g., the gateway and SP are EJBs). Indeed, Secure Service Façade does not require that the gateway and SPs are implemented using the same platform [128]. In case they are implemented using heterogeneous platforms, the gateway may extract the required identity (e.g., the username of the original client) from the security context and embed it in a SOAP header to the SPs.

6.3.2.3 Implementation

Developers have several implementations for a security gateway available. For examples, Forgerock implements a reverse proxy to filter traffic to and from the applications [146]. Alternatively, WSO2 offers an *Enterprise Service Bus* to support additional protections: (1) a monitoring and centralized auditing for security events, (2) an XML firewall to prevent XML Denial of Service, (3) an XML Schema definition to validate XML traffic that the gateway expects at runtime, (4) a caching of requests and responses that have the same hash value [145].

6.4 Conclusion

This chapter emphasised that an AAI implementation is programming language-dependent. It requires developers to have specialised security knowledge to understand various protocols, application containers, and proprietary APIs of an IdP that a cloud provider may support. Existing solutions provide developers with frameworks to adapt their applications manually. However, it forces developers to relearn and implement repetitive tasks using APIs from different cloud providers, instead of focusing on their business functionalities.

This chapter also explained several methods to propagate user identity between the application components. Among them, the identity propagation via HTTP headers is independent of any programming languages but has the following limitations. Firstly, administrators need to know the HTTP headers that an application component expects in order to configure the Web agent for passing them correctly. However, administrators may not have inside knowledge about the development of the application component. Secondly, the Web agent may propagate a number of HTTP headers that the application component may not

understand. As a result, developers have to adapt their application component to retrieve the correct HTTP headers from the Web agent and map them in the application context manually.

In general, various scenarios and protocols for propagating PII between SPs, which may confuse the application developers. However, at the end of the day, an application requires PII for the purpose of auditing and authorisation. Therefore, this chapter has searched for the authorisation design patterns that the developers frequently use to protect their applications. The thesis may describe these patterns as modelling templates so that an adaptation process could reuse them to auto-protect the applications and save the working effort of the developers.

This chapter also reviewed existing vendors in the industry that offer the solutions for IDM. They often provide their solutions as a whole package (e.g., their Web agent only work in their IDM system but is not compatible with the other vendors). Therefore, the research may consider the application as unprotected and migrate to another cloud provider. The Web agent component may not migrate together with the application but is provisioned by the target cloud provider so that it is compatible with the IDM system of the target cloud provider.

Chapter 7

Architecture design

Chapter 7 describes the architecture design of IDaaS. Section 7.1 first gives an overview of the thesis. Then Section 7.2 dives deeper in the concept of Purpose-based Encryption. Finally, Section 7.3 dives deeper into the concept of security infrastructure adaptation. Especially, Section 7.2.1 and Section 7.3.1 explain why researchers made their decisions on Purpose-based Encryption and security infrastructure adaptation, respectively.

7.1 Architecture overview

The following parts describe the main contributions of the thesis (Section 7.1.1) and give an overview of the reference architecture of IDaaS (Section 7.1.2).

7.1.1 Overview

The thesis has three main contributions summarised as follow:

1. Recall that none of the current design patterns of IDM can disseminate PII in federated security domains and protect PII at the same time (Section 3.1). For instance, the identity broker can federate user identity in various security domains but it relies on a single broker to be accountable and responsible for many parties in other security domains. To fix this missing gap, the research proposes a novel reference architecture for IDaaS, whereby multiple IdPs and SPs have defined roles and responsibilities for disclosing PII. The main goal is that if a dishonest entity does not follow the protocol, an honest entity can prove that, the other side has violated the protocol and the dishonest entity must pay for the penalty, according to GDPR. The next section will explain the proposed architecture of IDaaS in details.
2. Recall that many efforts in the past 10 years have been taken in privacy-preserving user identity. They target certain issues but still have limitations. For instance, users require interacting with the SPs over the frontend; none

of them protects identity propagation between intermediaries and against an untrusted host at the same time. To fix these issues, the thesis proposes a novel approach to Purpose-based Encryption (PBE). To the best of researchers' knowledge, this is the first approach to combine Purpose-based Access Control and Attribute-based Encryption to protect the confidentiality of disseminated data with multi-authorities support. Briefly, a user Bob encrypts his PII with a disclosure policy based on three main factors: "domains" (e.g., Facebook, Salesforce), "time" (e.g., 14 days), and "purposes" (e.g., marketing, purchase). Then the encrypted PII can be disseminated over untrusted hosts and intermediate SPs in a call chain. However, only SPs hosted in specific domains can decrypt the ciphertext within the given period, if and only if the access purposes of the service satisfy the intended purposes that Bob specifies before. After 14 days, SPs cannot decrypt the ciphertext, even if they were authorised to decrypt it before. Finally, it is worth mentioning that prior research in Purpose-based Access Control had difficulties in determining an access purpose of a request (Section 4.3), PBE solves this limitation by introducing a *purpose authorisation request*.

3. Recall that related work is missing a holistic approach to adapt the AAI implementations for cloud services in multiple cloud providers. None of them can control the identity propagation from the original caller to the end SP to complete a business transaction on demand. To the best of researchers' knowledge, this is the first approach that considers AAI implementation as a virtualization layer to govern the identity flow between SPs in a call chain and protects PII at the same time. In particular, the thesis considers the application components as *unprotected*. It moves the security mechanisms as well as the identity propagation out of the application implementation and

models them as a novel *security topology*. Briefly, the security topology describes the AAI components, their relationships with the application components in a platform-independent modelling language. Using the security topology, IDaaS can provision the AAI components, establishes a dynamic trust relationship between application components on demand, and evaluates any changes in the runtime environment to conform to the model. IDaaS also deploys the needed functionalities of PBE so that the AAI components can decrypt the ciphertext for the applications. At the end of the day, developers do not need to change the implementations of their applications.

7.1.2 Reference architecture of IDaaS

This section proposes the reference architecture of IDaaS. In general, the proposed architecture also follows federated IDM. However, it separates the IdPs in two roles with different responsibilities: a home IDaaS and a visitor IDaaS as in Figure 33.

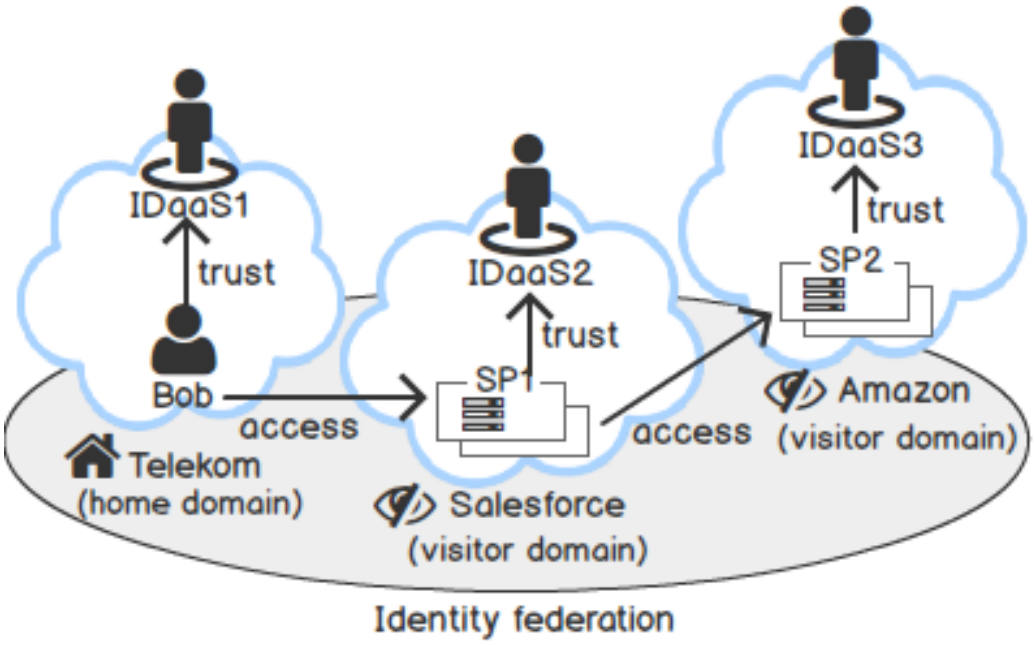

Figure 33: Reference architecture for IDaaS overview

Figure 33 shows an example of three security domains: Telekom, Salesforce, and Amazon. Each domain has an IDaaS. A user Bob registers at his *home IDaaS* (e.g., Telekom) for authentication. The IDaaS in the other domains (e.g., Salesforce, Amazon) federate with the home IDaaS. They are *visitor IDaaS*. A trust establishment should take advantage of existing trust relationships. In particular, users and SPs have signed contracts with their IDaaS, so they built *direct trust* relationships. From the user's perspective, Bob trusts his home IDaaS to collect and disseminate encrypted PII on behalf of him. From the perspective of SPs, IDaaS registers natural users and is accountable for their actions with a billing system.

In one security domain, all SPs trust its IDaaS as an authoritative resource to handle authentication requests (e.g., SP2 trusts IDaaS2, SP3 trusts IDaaS3). This thesis considers trust between users and an SP in the same domain as well as between federated domains as *indirect trust*. An automated trust negotiation between a user and an SP is the responsibility of the IDaaS and not of the SP. This trust model reduces the complexity for a user and an SP to establish trust with each partner individually.

To protect the dissemination of PII in federated domains, the home and visitor IDaaS also has different roles and responsibilities as in Figure 34:

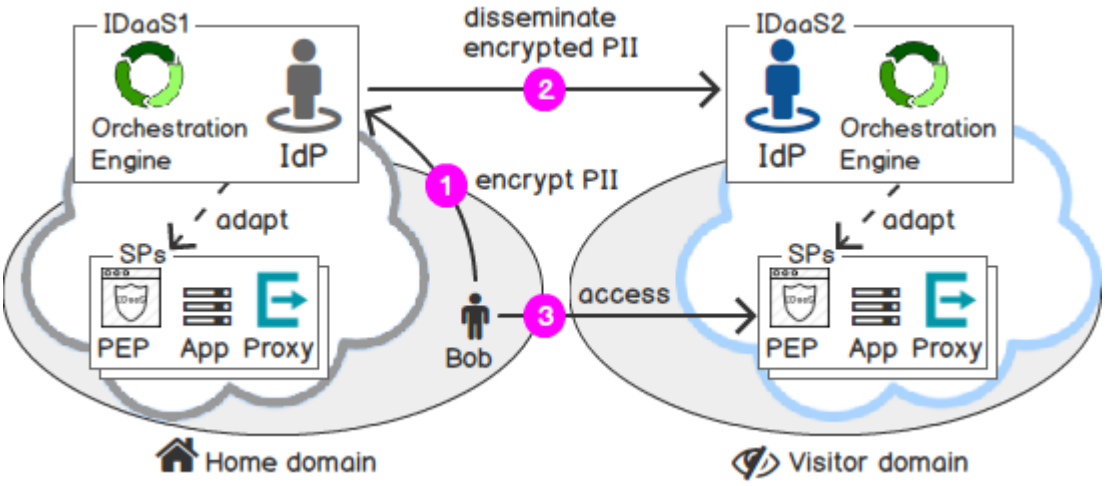

Figure 34: Roles and responsibilities of IDaaS

In Figure 34, a user Bob encrypts his PII and stores the ciphertext in his home IDaaS. Then the home IDaaS disseminates the ciphertext in federated domains. When Bob wants to access an SP in the visitor domain, he first authenticates to his home IDaaS and accesses the SP in another domain. While the home IDaaS is responsible for issuing a *time* authentication response, the visitor IDaaS is responsible for issuing a *purpose* authorisation response. In the concept of PBE, SPs use the time authentication and the purpose authorisation response to decrypt the ciphertext. Section 7.2 will describe the concept of PBE in more details.

In Figure 34, to govern the identity flow between SPs in a call chain, both the home and visitor IDaaS can adapt the AAI implementations for SPs in its security domain. Section 7.3 will describe the concept of security infrastructure adaptation in more details.

7.2 Purpose-based Encryption

The following sections dive deeper into the PBE concept. First, Section 7.2.1 describes the architecture considerations and explains why researchers made their decisions. Then, Section 7.2.2 and Section 7.2.3 summarise the ABE

scheme and use the given scheme to encrypt PII, respectively. Finally, Section 7.2.4 presents the request flow and the lifecycle of PII in federated domains.

7.2.1 Design principles for Purpose-based Encryption

To fulfil the research challenges in privacy-preserving user identity, researchers made the following decisions:

7.2.1.1 The decision on the confidentiality approach

In Section 4.2, the research reviewed two categories for privacy-preserving user identity: The first category preserves complete anonymity for users, and the second one protects the confidentiality of PII. While the first category satisfies the privacy properties undetectability and unlinkability, it does not support identity propagation in federated domains. Recall that identity propagation is necessary because each SP in a call chain requires user identity to complete a business transaction or to customise the service. This is the main reason why the approach in this thesis follows the second category but not the first one. In this case, it is acceptable to lose the property of undetectability because users trust an IDaaS, by definition, to complete a transaction.

7.2.1.2 The decision on the Purpose-based Access Control

In FIDM, when users disclose their PII over a front-end service, an intermediary SP in the back-end could be unknown them. Therefore, researchers reviewed PBAC as an alternative access control mechanism that depends on an intended purpose and an access purpose of a request (Section 4.3). In comparison to the traditional RBAC, PBAC is suitable for a large distributed and heterogeneous environment. By using PBAC, the disclosure of PII does not limit to a target SP or a specific local role. PBE uses PBAC as alternative access control and

follows GDPR by taking “time”, “purpose”, and “domain” as the main factors for describing the disclosure policy.

7.2.1.3 The decision on the ABE encryption scheme

Researchers borrow the idea of Mont *et al.* [81] to protect PII against untrusted hosts by binding PII with the disclosure policy using an encryption scheme. In Section 4.5, researchers found two alternative encryption schemes: ABE and IPE. They choose ABE over IPE for two reasons: Firstly, the intended purposes are public information. Therefore, it is not necessary to hide them in the attribute index. Secondly, the public index is faster, and the access policy is much more expressive than the private index [106]. In ABE, researchers also choose CP-ABE over KP-ABE because users know the intended purposes of collecting user data beforehand and encrypt the data with an access policy accordingly. Finally, researchers choose the ABE scheme of Rouselakis and Waters [115] as it supports a large universe of attributes and multi-authority at the same time.

7.2.1.4 The decision on the home and visitor authority

The “seven laws of identity” [14] recommends not to limit users to a single authority. Thus, in the reference architecture of IDaaS, users can encrypt and distribute their encrypted data in any security domains. However, only SPs that satisfy the disclosure policy can obtain the key capabilities to decrypt the data.

According to GDPR, the data controller is accountable for transferring PII to the data processor. That is, the data controller must determine the purposes and types of the data processor. In other words, when a cloud service registers on a cloud provider, it must provide its business information to the cloud provider such as the access purpose of the service. IDaaS could use this information to authorise a purpose request for accessing PII in its domain. In particular, the

home IDaaS issues a secret share about the user authentication time in the home domain. The visitor IDaaS issues a secret share about the access purpose of the relying SP in the visitor domain. The SP in the visitor domain can then decrypt the ciphertext if it has both shares that satisfy the disclosure policy. In such a case, the SP is authorised to access PII for the given purposes and in the allowed period. Any parties that violent the protocol can be identified and have to pay for the penalty.

In Figure 35, the home IDaaS (e.g., Telekom) disseminates the encrypted PII on behalf of user Bob to federated domains (step 1). On demand, Bob authenticates to his home IDaaS, which enables him to access multiple SPs in federated domains (step 2, 3, and 4).

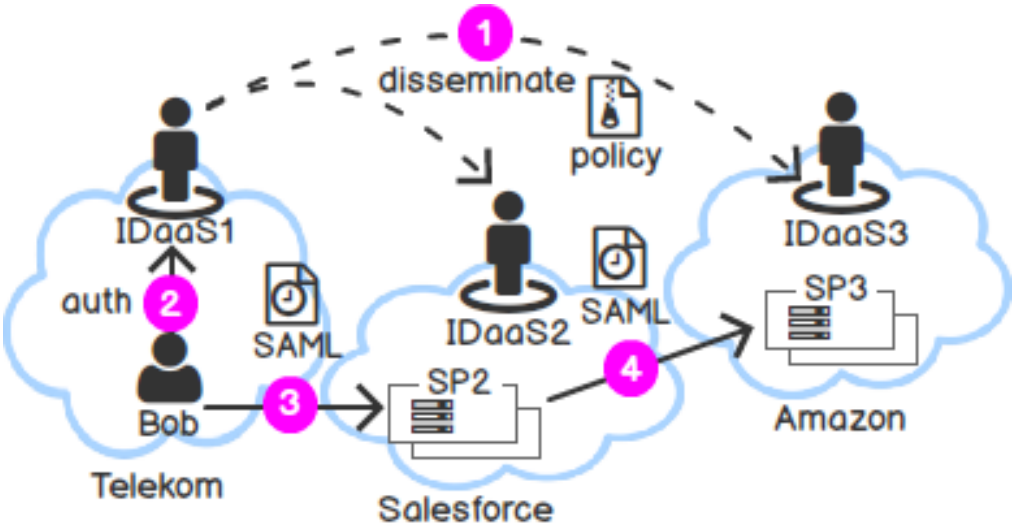

Figure 35: Purpose-based Encryption overview

7.2.2 Multiple-authority ABE scheme

This section summarises the multi-authority ABE scheme in [115]. In this scheme, multiple authorities can issue secret keys associated with the attributes under their control independently. The scheme consists of the following phases:

1. *AuthSetup (PID, GP) → {PK_{PID}, MSK_{PID}}*: The authority setup algorithm takes a provider identifier *PID* and the global parameters *GP* as inputs. It outputs a public key *PK_{PID}* and a master secret key *MSK_{PID}* for the given authority.
2. *KenGen (UID, SESSIONID, MSK_{PID}, U, GP) → $SK_{UID,U,SESSIONID}$:* The key generation algorithm takes in a user identifier *UID*, a transaction identifier *SESSIONID*, a master secret key MSK, an attribute *U* (controlled by the authority), and the global parameters. It outputs a secret key associated with the attribute *U* for the user *UID* in the transaction *SESSIONID*.
3. *Encrypt (M, P, {PK_{PID}}, GP) → CT:* The encryption algorithm encrypts a message M with a disclosure policy P, the public key PK_{PID} of the relevant authority, and the GP. It outputs the ciphertext CT.
4. *Decrypt (CT, {$SK_{UID,U,SESSIONID}$}, GP) → M:* The decryption algorithm takes a ciphertext *CT*, a set of secret keys and the global parameters. It outputs the message *M* if the attributes in the secret keys satisfy the disclosure policy *P*. Otherwise the decryption fails.

In comparison to the ABE scheme in [115], the key generation uses the *UID* concatenating with the *SESSIONID* as an input for tying all key attributes together. The key generation adds a *SESSIONID* due to the following reason. The ABE scheme prevents collusion attacks between users (i.e., different users cannot combine their keys to gain more capabilities for decryption). However, the scheme does not prevent various entities from combining secret keys about the same user. This situation happens when an SP in a call chain receives a secret key about a user and may try to collude with other services in a different transaction. In such a case, the decryption also fails if SPs try to combine secret keys from various transactions. The thesis calls this input a “linchpin” for tying all key attributes of the same user in the same session.

7.2.3 PII Encryption

This section describes how to encrypt PII using the above ABE scheme. Figure 36 shows an example, whereby PII (right column) is encrypted with a disclosure policy (left column).

```
1  # INDEX                              # PAYLOAD (ciphertext)
2  "id1"
3  AND 1 OF ("current")                  <user data>
4  AND 1 OF ("purchase", "delivery")
5  AND 1 OF ("salesforce", "amazon")
6  AND "eu"
7  AND "year2018"
8  AND dayaccess IN (250, 264)
```

Figure 36: An example of PBE

In this example, the data owner “id1” encrypts his data with an intended purpose to complete the “current” transaction (line 3), in particular, “purchase” or “delivery” a product (line 4). The PII is available for SPs managed by the IDM of “salesforce” or “amazon” and hosted within the “eu” country (line 5 and 6). The ciphertext is valid within 14 days (line 7 and 8 present a range of dates from 07.09.2018 to 21.09.2018). Also, the home IDaaS optionally signs the payload of the ciphertext.

In general, the disclosure policy uses the following attributes. Among them, the solution concept borrows *purpose*, *ppurpose* (i.e., primary purpose) from the Platform for Privacy Preferences Project (P3P) [17]:

- *uid:* a pseudonym that identifies the data owner known only to the home IDaaS.
- *purpose:* specifies an intended purpose for disclosing this PII (e.g., “current”, “tailoring”, “pseudo-analysis”, and “telemarketing”).
- *ppurpose:* if the *purpose* is set to “current”, use this element to provide more details about the current business transaction (e.g., “purchase”, “login”).

- *domain:* specifies a security domain for disclosing PII. SPs outside the specified domains are not allowed to process the PII.
- *country:* specifies the location of the servers that process the PII. SPs hosted outside this country are not authorised to process the PII.
- *access time:* limits the access time of a user request. The access time is valid within the creation time and the expiration time of the ciphertext.

Generalisation

Generalisation is part of the encryption process to prevent disclosing an exact value for a particular purpose before encrypting it (e.g., the process generalises a user salary with minimum and maximum values to hide the exact value from "marketing" purpose).

Minimisation

Minimisation is part of the encryption process to prevent disclosing more user data than the stated purpose. For example, the P3P specifies that the purposes "develop", "pseudo-analysis", and "pseudo decision" should not include any personal information to target, profile, or contact the individual. Therefore, the minimisation process makes sure, for instance, "name" and "email" are not included in the payload before encrypting it.

7.2.4 Request flow of PBE

The following part revisits the scenario where a user Bob from Telekom consumes SPs in Salesforce. Briefly, Telekom authenticates Bob and issues a *Time Access Token* (TAT). Salesforce is in charge of the purpose authorisation request and issues a *Purpose Access Token* (PAT) for SPs in its domain. An SP can decrypt the PII if it possesses both the time and purpose token. This results in the following request flow as in Figure 37. It worth mentioning that steps 2, 4,

and 5 represent the standard request flow of Attribute-based Access Control [6]. The concept extends the standard request flow in step 1 and 3.

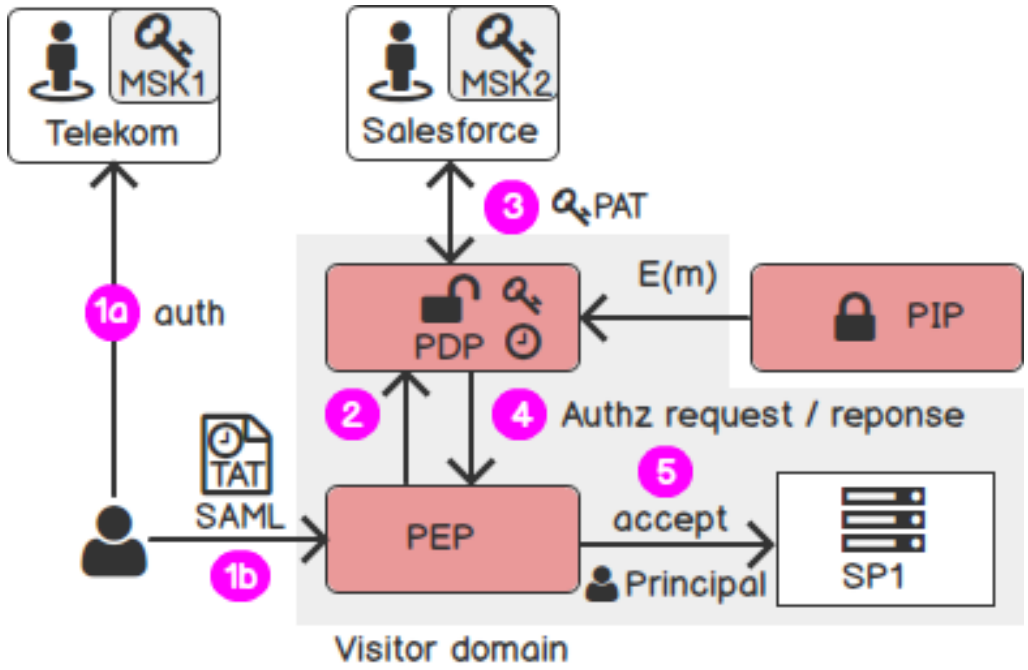

Figure 37: Request flow of PBE in multi-authority

1. **Encryption**: Bob encrypts his data as described in Section 7.2.3. In particular, Bob uses the public key of Telekom to encrypt the data with the policies “access time”, “domain”, and “country”. He uses the public key of Salesforce to encrypt the data with the policies “purpose” and “ppurpose”. The algorithm outputs one ciphertext for each user attribute and disseminates it to Telekom.
2. **Authentication**: Bob accesses SP1 and is redirected to Telekom for authentication. Telekom authenticates Bob and issues a SAML response (step 1a). Telekom uses the key generation algorithm to issue a TAT. This token consists of three key attributes: “access time”, “domain”, and “country”. The key generation also uses the user identifier and the session identifier (in the SAML response) as a “linchpin” for tying all key attributes together. Telekom includes the TAT in a SAML response and returns to the user. In

addition, the IdP uses the public key of SP1 to encrypt the TAT so that only SP1 can decrypt it.

3. **Authorisation**: Bob gives the SAML response to access SP1 at Salesforce (step 1b). The PEP validates the SAML for authentication (not shown in the figure) and forwards the SAML to the Policy Decision Point (PDP) for authorisation (step 2). The PDP again submits the SAML to the Salesforce's IdP to exchange for a PAT (step 3). Salesforce uses the key generation algorithm to issue a PAT with two key attributes "purpose" and "ppurpose". It also uses the user identifier and the session identifier delivered in the SAML as a "linchpin" for tying all key attributes together.
4. **Decryption**: Now the PDP has enough key attributes. It combines the TAT with the PAT. If all key attributes satisfy the disclosure policy and the "linchpin" is the same, the PDP can decrypt the ciphertext. Otherwise, the decryption will fail. After the PDP decrypts the PII, it returns the user attributes to the PEP and to the application in the backend (step 4 and 5).

7.3 Security infrastructure adaptation

The following sections dive deeper into the adaptation concept for the security infrastructures. Section 7.3.1 first explains the decisions on using the security topology. Then Section 7.3.2 models the security topology for the adaptation process.

7.3.1 Design principles for security infrastructure adaptation

To solve the research challenges of the security infrastructure adaptation, researchers made the following decisions:

7.3.1.1 The decision on security topology

The research has searched for a holistic approach to adapt applications to the cloud environment (Section 3.2.1). What researchers have learned from recent work is how important it is to understand the application topology. Because the implementation of an application is like “black box” for the adaptation process, so the first step is to decompose an application in a topology view. The topology view should describe which application component is hosted on which compute node and in which network so that the adaptation process can get the information from the underlying network and the compute node at runtime (e.g., IP address, network port, etc.). This runtime information is necessary for the deployment of the security components to protect the application components endpoint in the backend. Also, the topology view should describe which application components connect to each other internally. This information is necessary to configure the security components to govern the needed identity flow between the (internal and external) application components on demand.

The adaptation process in this thesis also follows the holistic approach to cloud adaptation. However, to support the reverse engineering phase, it requires a security architect describing the topology views. In particular, the approach considers a separation of duties between a *security architect* (i.e., who is an expert in security and understands the application topology), an IDaaS, and application developers. First, a security architect describes the security infrastructure (i.e., the AAI components and the relationship of the AAI components with the applications) and expresses his requirements (the “what”). Then, the IDaaS provision the security infrastructure to protect the application according to the requirements (the “how”).

In this approach, the security infrastructure decouples from the application logic. Thus, the security implementation may change without affecting the application. The provisioning of security infrastructure may depend on proprietary APIs and functionalities of a target cloud provider. Because every cloud provider may have its implementation of the underlying infrastructure (e.g., perimeter firewalls, intrusion detection, or a vendor-specific IdP), they have the best knowledge to provision the security infrastructure of the application and adapt it to their underlying infrastructure appropriately.

7.3.1.2 The decision on identity propagation

Recall in chapter 5, the implementation of identity propagation between SPs is a time-consuming task. Therefore, IDaaS also adapts the underlying security infrastructures to propagate the needed PII between cloud services as in Figure 38. In the end, application developers do not need to implement how they propagate PII between cloud services to complete a business transaction. They only need to care about what PII is available to them from the environment so they can use it in their applications.

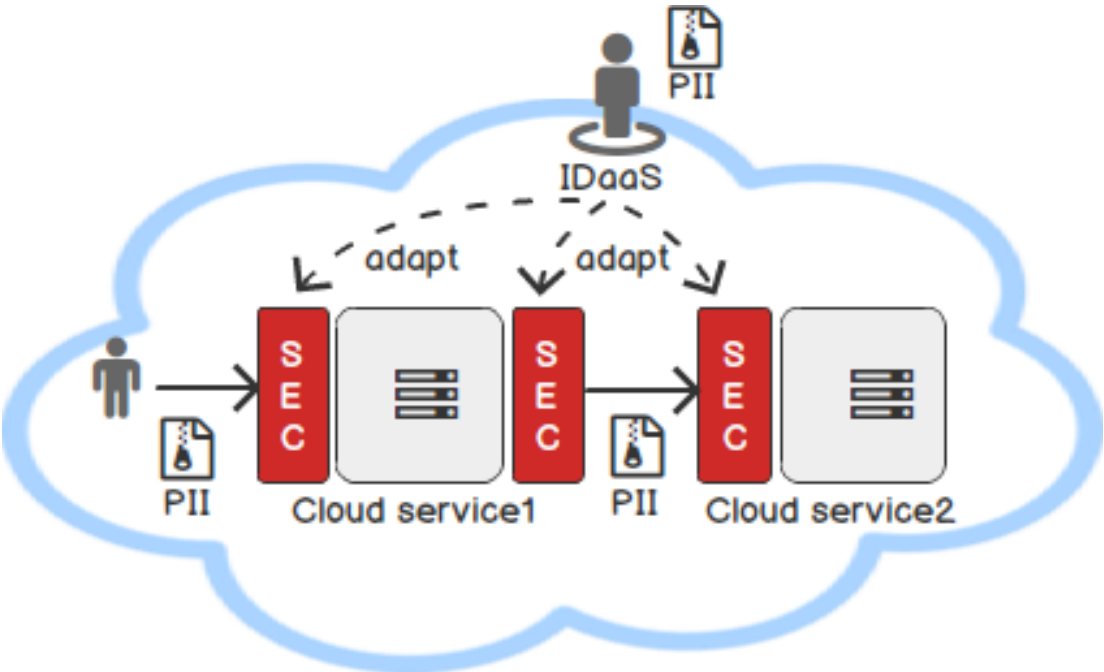

Figure 38: IDaaS adapts the security infrastructures to propagate PII.

For the identity propagation to work, the security architects have to express the requirements (in the security topology), which PII their applications expect so that IDaaS can adapt the security components to propagate it accordingly. Section 6.3.1 also showed that multiple programming languages propagate PII in different ways. Because the security architects understand their applications the most, they can specify how an identity propagation should occur. For example, they may define that the security component should propagate an email address from the user in an HTTP header. Otherwise, the application components in the back-end cannot use the propagated PII. For this reason, the security topology needs to define *identity mappings* between the security and the application components. By including identity mappings in the security topology, the adaptation process can understand the identity flow between the components and solves the interoperability issue between them.

Figure 38 also illustrates that PII is propagated in ciphertext format between intermediary services. IDaaS deploys the security components with the functionalities of PBE so that they can understand the PBE protocol to decrypt the ciphertext automatically and propagate the needed PII in plaintext to the application components in the backend.

Figure 39 shows the adaptation process of the security infrastructure.

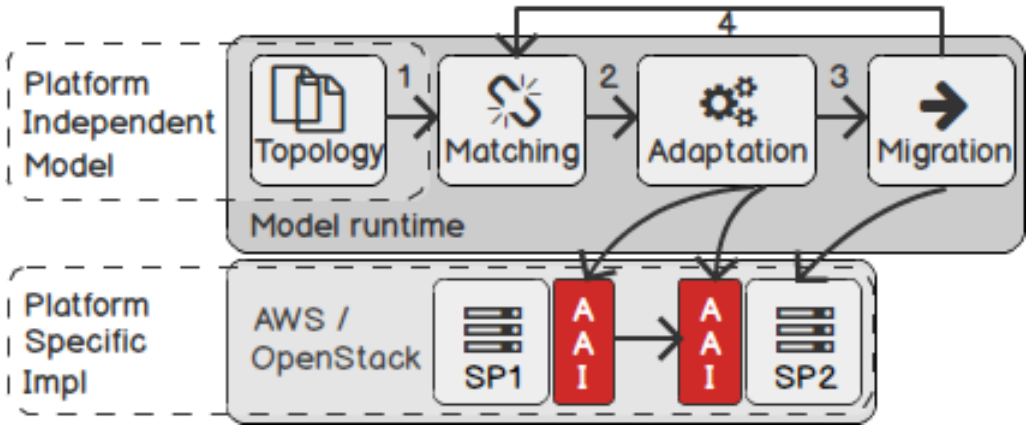

Figure 39: The adaptation process of the security infrastructure

The adaptation process starts by registering the security topologies of cloud services in a modelling engine (i.e., an orchestration engine). The topology describes all trust capabilities (and requirements) of AAI components. When two cloud services sign a B2B contract, the orchestrator matches their trust capabilities and requirements (step 1), then adapts their AAI implementations in a target cloud hosting (step 2). When a cloud service (e.g., SP2) migrates to another cloud provider (step 3), the adaptation process starts again with input from the new runtime environment (step 1). In general, it is not necessary to define a static trust between services, but the orchestrator establishes a dynamic trust between them.

7.3.1.3 The decision on how to enforce authentication and authorisation

Recall in Section 3.2.2, the AOP approach enforces authentication and authorisation for an application program on critical points (i.e., a component, a class, or a class method) but slows down the application performance. For this reason, IDaaS enforces authentication and authorisation at the APIs of an application component. To achieve this goal, the research has investigated how to decouple the authentication and authorisation from the application implementation. As pointed out in Section 6.3, the decoupling of the security implementation outside the Web application also increases the application performance, because the Web server now performs the security-related tasks [128].

To save the working effort of developers, researchers have collected the most frequently used security design patterns (Section 6.2) and then model the security components as topology templates. As a result, the security architects can reuse the topology templates in describing the security topology. The next section describes how to model the security topologies in details.

7.3.2 Security topology

An *application topology* is a description of all application components in the application (e.g., a Web application, a database), their relationships (e.g., the Web application "connects to" the database), and how to deploy these components. The Topology and Orchestration Specification for Cloud Applications (TOSCA) [147] is a standard specification to describe an application topology. In TOSCA, components are defined as *node types*. Relationships between components are defined as *relationship types*. A component may publish some information for other components to establish a relationship with it as *capabilities*. The following sections define the novel types for the security components, the "protect" relationship between the security components and the application components, as well as the "trust" relationship between the security components.

7.3.2.1 Node types

Figure 40 shows the node types for security components. They are "PolicyEnforcementPoint" (PEP), "PolicyDecisionPoint" (not shown in the figure), and "Proxy".

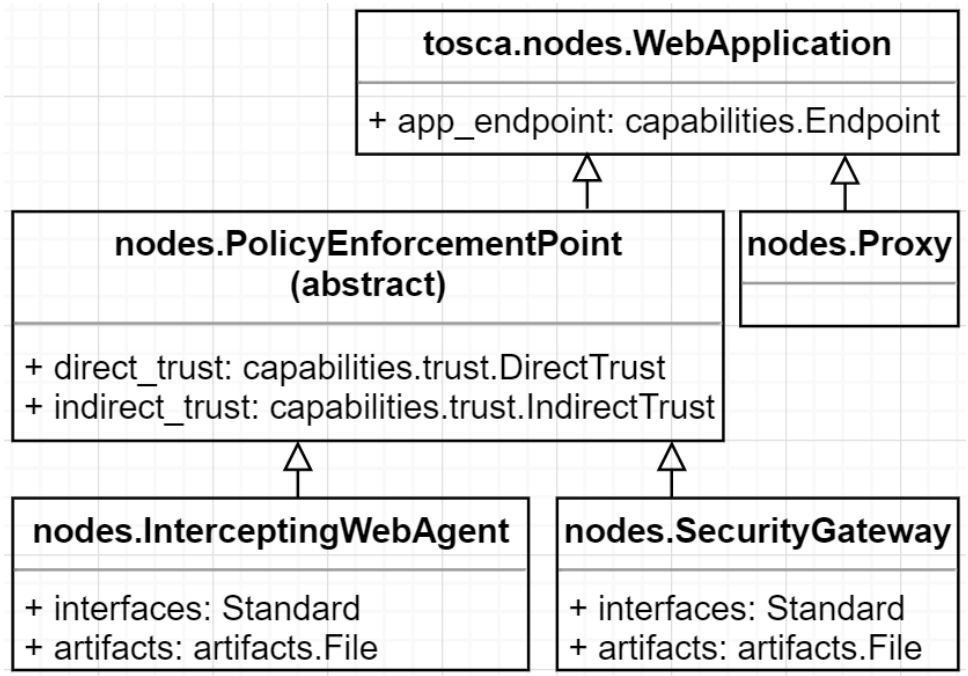

Figure 40: Node types for security components

These nodes derive from the normative node type “WebApplication” of TOSCA, thus they have an application endpoint capability by default. This capability describes useful information such as protocol, port, and a network of an application component.

The PEP node is an abstract node type with trust capabilities (e.g., direct trust). These trust capabilities provide the necessary information for a service consumer to establish a trust relationship with an SP. The PEP does not have any implementation but offers two sub-nodes: an “InterceptingWebAgent” and a “SecurityGateway”. A security architect can choose one of these nodes to protect a Web application in the backend.

The Proxy node is an outbound gateway for a service consumer to connect to an SP. A security architect uses the Proxy to intercept outbound requests and to enforce additional security mechanisms to them before calling the PEP of the SP. In TOSCA, all node types have *implementation interfaces* and *deployment*

artifacts. The orchestration engine will call the implementation interfaces and use the deployment artifacts to provision the nodes.

7.3.2.2 Trust capabilities

In TOSCA, a relationship is a directional connection from a source node to a target node. A target node can declare *capabilities* as a set of data that it exposes to a source node in a relationship. On the other hand, a source node can declare *requirements* that a node needs to fulfil to be instantiated. The trust relationship between a service consumer and an SP is modelled as follows:

From the server's perspective, an SP wants to define whom it can trust for providing service. Recall that a trustor trusts a trustee on a specific context when the trustee can show validated attributes that satisfy the trustor's policy [2]. In this case, SP is the trustor, the service consumer is the trustee, and the service consumer shows validated attributes that satisfy the SP's policy. Figure 41 shows the trust model.

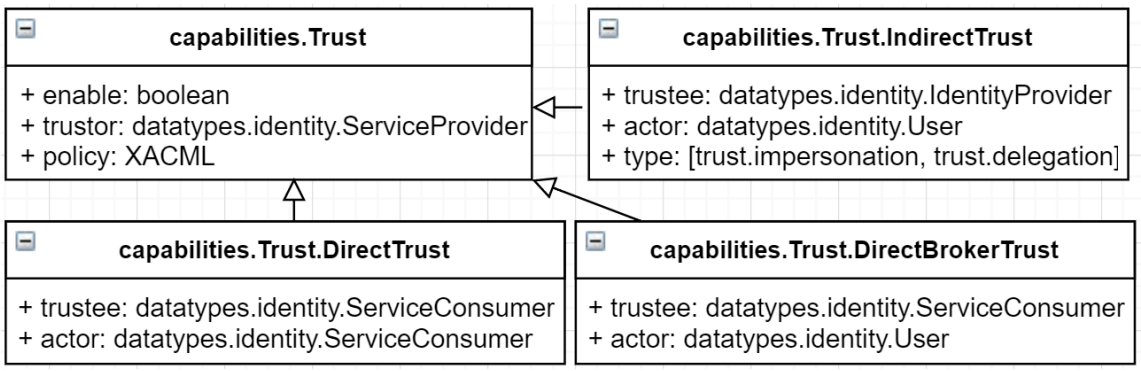

Figure 41: Trust capabilities

In Figure 41, the "capabilities.Trust" is an abstract trust model: a *trustor* (i.e., an SP) trusts a *trustee* (to be defined) to delegate permissions for an *actor* subject when the actor subject satisfies the trustor's security policy. If the trust evaluation is correct, the SP continues to execute under the actor's identity. The abstract trust model has three sub capabilities:

1. **Trust.DirectTrust**: An SP (trustor) trusts a service consumer (trustee) directly. When using this trust model, the service consumer indicates the SP to execute further using the service consumer's identity itself (actor). This trust model generates a security policy that builds a mutual trust relationship between Web services such as the *SSL transport binding* [148].
2. **Trust.DirectBrokerTrust**: An SP (trustor) trusts a service consumer (trustee) to generate a self-signed identity for a user. The service consumer indicates the SP to execute further using the user identity (actor). This trust model generates a security policy that builds a mutual trust relationship between Web services and transmits a self-signed user identity such as the *SAML assertion Sender Vouches over SSL* [148].
3. **Trust.IndirectTrust**: An SP (trustor) trusts an IdP (trustee) for issuing a user identity. The service consumer may act on behalf of a user (i.e., identity impersonation) or act as the user (i.e., identity delegation) to request a user identity from the IdP. The consumer indicates the SP to execute further using the user identity (actor). This trust model generates a security policy that establishes trust with an unknown Web service and transmits a user identity such as the *symmetric binding with supporting token* [148].

In summary, by using these trust capabilities, it is possible to express both direct trust and indirect trust relationships between a service consumer and an SP (Section 6.2) and to cover all scenarios associated with identity propagation (Chapter 5).

7.3.2.3 Data types

In the trust capabilities, the trustee and actor have a data type that gives more information about a given entity. For instance, the data type "identity.IdentityProvider" has an element "sts_wsdl" to describe the STS

endpoint of the IDaaS. The data type “identity.User” has two elements “claims_mapping” and “group_mapping”. These data types allow the security architect to express which identities his application needs from the runtime environment (more details in the experimental results).

7.3.2.4 Relationship types

The protect relationship

Figure 42 shows how to model a “protect” relationship between an “InterceptingWebAgent” node and a “WebService” node.

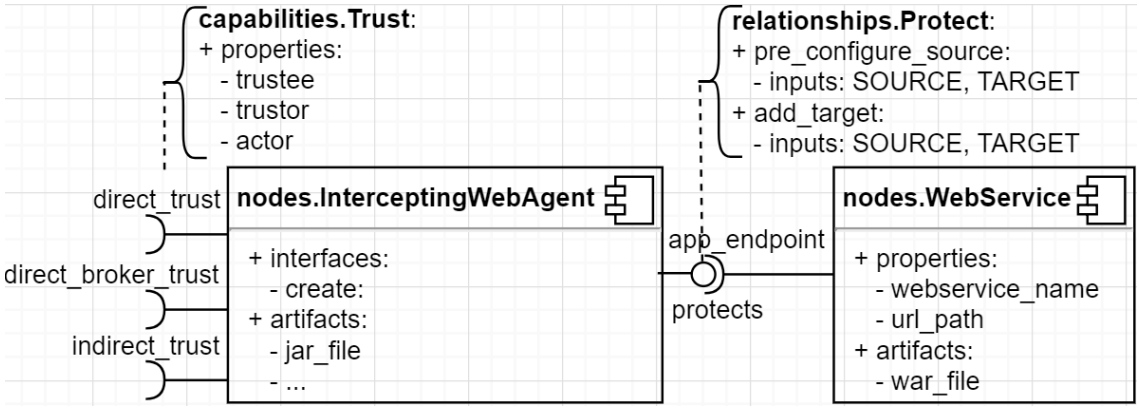

Figure 42: Protection relationship

The “protect” relationship derives from the normative relationship “ConnectsTo” of TOSCA. This relationship defines a connection between a source node and a target node. In this case, the “InterceptingWebAgent” is the source node, and the “WebService” is the target node. The relationship also provides standard interfaces such as “pre_configure_source” and “add_target”. During the deployment, the orchestration engine will call these interfaces to configure the relationship between them.

The trust relationship

Figure 43 shows a topology of a service consumer for outbound traffic. From the client’s perspective, the service consumer may or may not propagate a user identity to the SP.

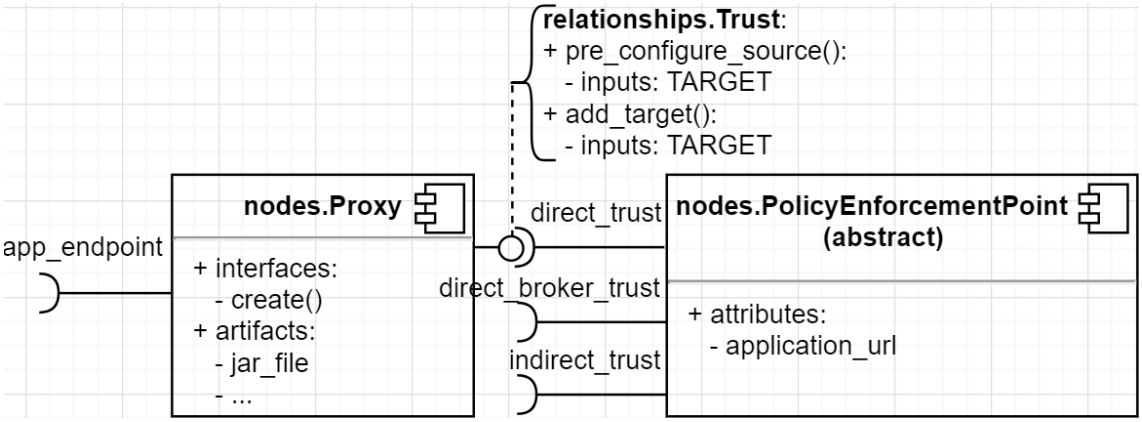

Figure 43: Trust relationship

In Figure 43, the security architect can choose if his Proxy shall connect to the PEP (of the SP) via a trust capability. Here he indicates to which degree the SP obtains a user identity and how they may use it (more details in the experimental results).

7.3.2.5 Substitution

In the topology of the service consumer, PEP is a *substitution node* that represents the other endpoint of an SP. Topology substitution is a powerful concept of TOSCA. It considers an application topology as a single reusable node. At runtime, the SP exposes its endpoint and trust capabilities in the PEP node. Then the service consumer can read this information from the PEP node and does not need to understand the complexity of the SP's underlying system.

7.4 Conclusion

The proposed architecture of IDaaS follows the federated IDM but unlike the identity broker, multiple IdPs and SPs have defined roles and responsibilities for disclosing PII. The home IDaaS issues a secret share about a user authentication time in a SAML response. The visitor IDaaS issues a secret share about an access purpose of an SP in an OAuth authorisation response. The SP can decrypt the ciphertext if it follows the protocol to have both shares

that satisfy the disclosure policy. In such a case, the SP is authorised to access PII for the right purposes and in the allowed period. Therefore, this approach is compliant with GDPR.

To complete the whole picture, both the home and visitor IDaaS can provision the required security components and govern the identity flow for the cloud services in its domain on demand. Because the implementation of an application is like a black box, IDaaS requires the security topology to understand the application (i.e. the runtime information of network and compute node, the trust and protect relationship between the security components and the application components, and the identity mapping between them) so that IDaaS can adapt the required security components accordingly.

Chapter 8

Experiments, Implementation, and Evaluation

This chapter describes the implementation and evaluates the concepts of PBE and the security infrastructure adaptation in Section 8.1 and Section 8.2, respectively.

8.1 Purpose-based Encryption

The following sections describe the experiments of PBE. Section 8.1.1 describes the test environment. Section 8.1.2 then explains the experimental results, and Section 8.1.3 describes the implementation of PBE. Finally, Section 8.1.4 evaluates the performance and security considerations of PBE.

8.1.1 Test environment

Figure 44 describes the test environment with a home domain and a visitor domain. Each domain has an IdP implemented by the open-source WSO2. In the home domain, researchers have developed a client application that is responsible for encrypting and storing PII in the home IDaaS. In the visitor domain, researchers have developed two Web applications to simulate a B2B relationship: The shopping service receives purchase orders from the user and then calls the delivery service for shipping the parcel.

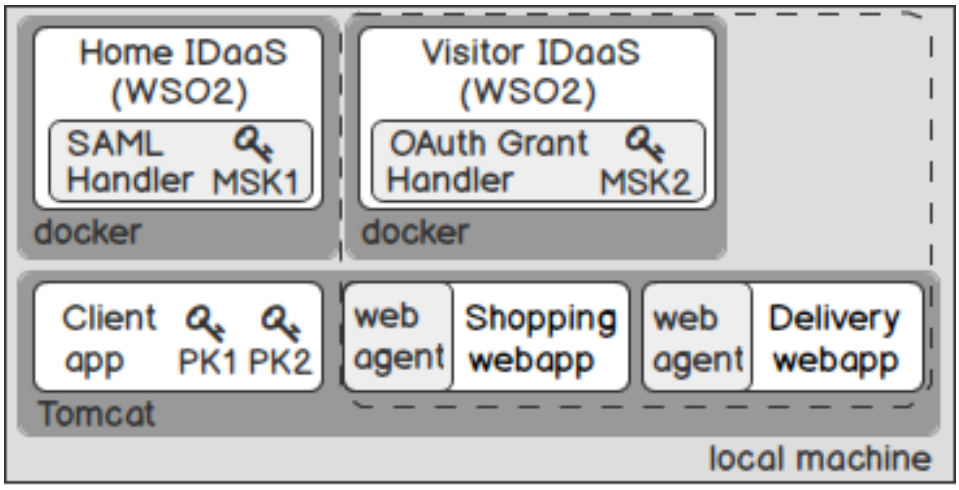

Figure 44: Test environment

In the proof of concept, researchers want to show that existing IdP vendors and applications can easily adapt their solution (i.e., existing applications do not need to change the implementations). Therefore, they have developed the following “handlers” that can be integrated into an existing system. In the home IDaaS, they have developed an OSGi bundle “SAML Handler”, which embeds a TAT in an authentication response on the fly. In the visitor IDaaS, they have developed an OSGi bundle “OAuth Grant Handler”, which embeds a PAT in an authorisation response on the fly. They have implemented a Web agent, which intercepts and handles the tokens. The Web agent also decrypts the ciphertext and propagates the PII in the payload to the Web application in the backend.

In addition, the test environment has the following key distribution for the components. For encryption, the client application requires the public keys of the home and visitor IDaaS (e.g., PK1, PK2). The “SAML Handler” requires a master secret key of Telekom (e.g., MSK1) for issuing the TAT. The “OAuth Grant Handler” requires a master secret key of Salesforce (e.g., MK2) for issuing the PAT. The Web agents do not require any keys from the ABE scheme for decrypting user data.

To isolate the test environment, researchers test each IDaaS inside a docker container separately. The Web applications run inside the local server Tomcat.

8.1.2 Experimental results

In the following section, researchers test the lifecycle of PII. They show that the shopping service and the delivery service can only decrypt a portion of PII to complete a purchase order and to deliver a parcel in 14 days, respectively, but nothing more.

1. **Encryption**: Listing 9 shows some examples of user identities under test.

Attribute name	Attribute value	Purpose	PPurpose	Domain	CN	Lifetime (days)
Last name	Vo	current, contact	purchase, delivery,	salesforce, amazon	eu	30
DOB	01.01.1980	current	purchase, government	salesforce, amazon	eu	30
Addresses	Berlin…	contact	delivery	salesforce	eu	14

Listing 9: Sample user data

In the given example, the user is willing to disclose his "date of birth" (DOB) for purchasing a product (e.g., in practice, a shopping service may use a birthday to confirm that a user is an adult). However, for delivering a parcel, he does not need to disclose his birthday. Furthermore, he only discloses his "addresses" in a "current" transaction and limited within 14 days after "delivery". In contrast to his "addresses", his "last name" has a longer lifetime (e.g., 30 days) and can be used for contacting purpose without having a "current" transaction (e.g., a marketing service may "contact" him later on for advertising a new product). Figure 45 shows the encryption result in WSO2, whereby PII is stored in some Based64-encoded ciphertext format.

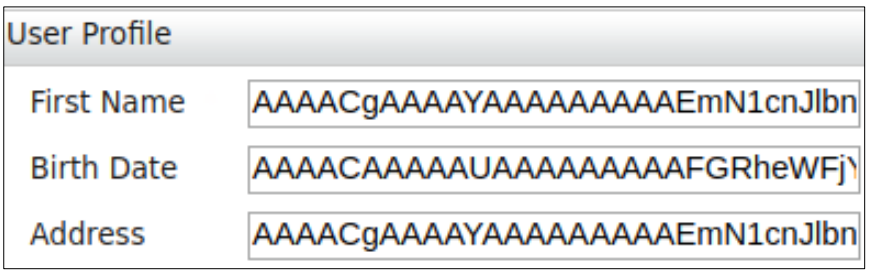

Figure 45: The user profile in WSO2

2. **Authentication**: Listing 10 shows an authentication response from the home IdP to the shopping service.

```
1  <Response Destination="http://.../shopping.com/home.jsp">
2   <Issuer>localhost</saml2:Issuer>
3   <Assertion>
4    <Subject>
5     <NameID>tri</saml2:NameID>
6    </Subject>
7    <AuthnStatement SessionIndex="db0940af-6f7d-485d-..."
8                    AuthnInstant="2017-10-16T14:05:24.415Z"
9                 SessionNotOnOrAfter="2017-10-16T14:20:24.415Z">
10   </AuthnStatement>
11   <AttributeStatement>
12    <Attribute Name="http://wso2.org/claims/dob">
13     <AttributeValue>AAAACAAAAAU...</AttributeValue>
14    </Attribute>
15    <Attribute Name="http://wso2.org/claims/addresses">
16     <AttributeValue>AAAACgAAAAY...</AttributeValue>
17    </Attribute>
18    ...
19    <EncryptedAttribute>
20     <!-- time access token -->
21    </EncryptedAttribute>
22   </AttributeStatement>
23  </Assertion>
24 </Response>
```

Listing 10: SAML Response with Time Access Token

In the SAML response, the home IdP issues several attribute statements about the user (lines 12 to 18). These attributes are in ciphertext format so the shopping service cannot read them (yet). The SAML response also contains the user identifier (line 5), the session identifier (line 7), and the expiration time of the session (line 9). The "SAML Handler" generates the TAT, encrypts it with the public keys of the audience restrictions (i.e., the shopping service and delivery service), and embeds the encrypted TAT in the SAML (lines 19 to 21).

3. **Purpose authorisation request**: After receiving the SAML response, the shopping service uses its private key to decrypt the TAT. Listing 11 shows the TAT in plaintext.

```
1  <Attribute Name="http://wso2.org/claims/timeAccessToken">
2   <AttributeValue>
3    dayaccess549@telekom:Abndy+/AzSNUYM6dZrr30/2kop9/CD...;
4    dayaccess237@telekom:HemLPJ3c4Y4//+jtfKruukl9upnLmU...;
5    ...
6    year2018@telekom:EQlR2W63klkqvJKh8ag2JV46gglsSSaShm...;
7    salesforce@telekom:Ax4xQBous+tU2VCrSSYNZE1SYiL7Q9xK...;
8    eu@telekom:FqqyIaOlFZJstzddoUnYDiNSG3heIzcb32/E/EZ6...;
9   </AttributeValue>
10 </Attribute>
```

Listing 11: Time access token

The TAT is a Based64-encoded format that contains some cryptographic key attributes about the current access time (lines 3 to 6), which is the current date that the user authenticates to his home IdP (see the implementation for more details). Also, the TAT has the key attributes issued for the visitor IdP "salesforce" with the domain "eu" (line 7 and 8).

Now the shopping service uses the SAML response to exchange for a PAT from the visitor IDaaS. To implement the purpose authorisation request, researchers extended the protocol SAML 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants [71] as in Listing 12.

```
1  POST /oauth2/token HTTP/1.1
2  Host: localhost:9443
3  Content-Type: application/x-www-form-urlencoded
4  Authorization: Basic BASE64(<client_id>:<client_secret>)
5
6  grant_type=urn:ietf:params:oauth:grant-type:saml2-bearer
7  &assertion=BASE64(<samlResponse>)
8
9  {
10   "access_token": "<purpose access token>",
11   "token_type": "bearer",
12   "expires_in": 3600,
13 }
```

Listing 12: Purpose authorisation request

The purpose authorisation request uses the "client_id" and "client_secret" of the shopping service (line 4) to authenticate to the visitor IDaaS. The POST request contains the SAML response in the "assertion" element (line 7), and the "grant_type" is set to "saml2-bearer" (line 6). After receiving the request,

the “OAuth Handler” generates a PAT and embeds it in the “access_token” element (line 10) of the response.

To issue the PAT, the “OAuth Handler” checks the expiration time of the SAML session. If the session is still valid, it issues a PAT containing two key attributes “current” and “purchase”. It means the shopping service is authorised for processing PII to “purchase” a product in the “current” transaction.

4. **Decryption**: Now the shopping service has both the TAT and PAT to decrypt the ciphertext. Figure 46 shows the GUI of the shopping service. The service can decrypt PII for the “purchase” purpose (e.g., given name, last name, birthday, and email address) but cannot decrypt the user “addresses”.

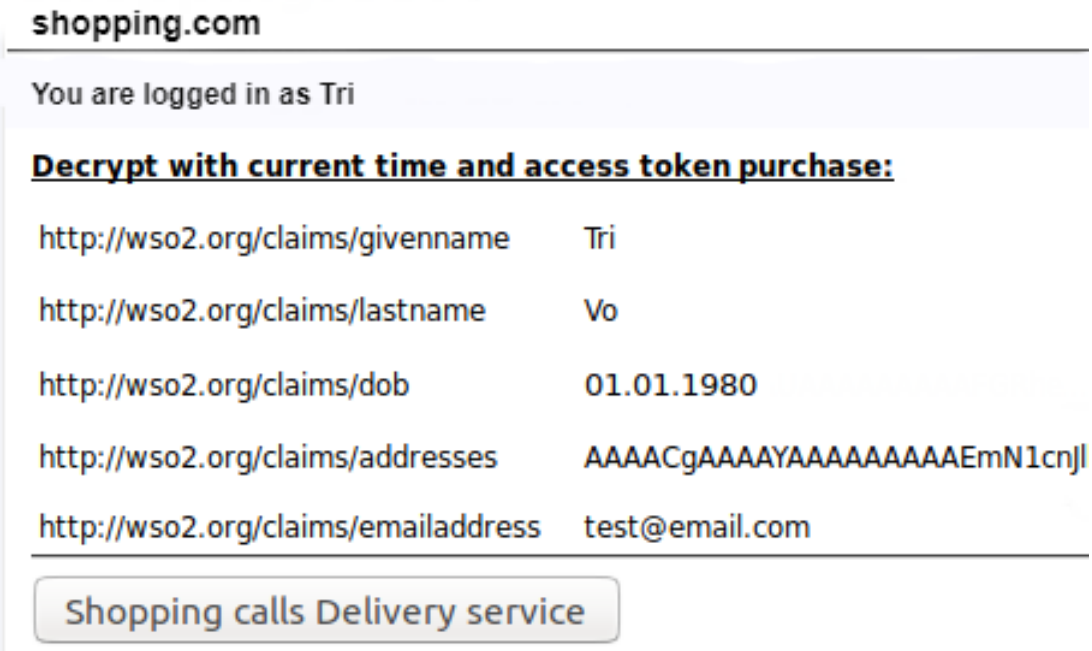

Figure 46: The shopping service GUI

5. **Identity propagation**: In this step, the shopping service requests the delivery service for sending the parcel. In this scenario, researchers assume that the delivery service trusts the shopping service for user authentication (i.e., a trusted subsystem [122]). Therefore, the shopping service only needs to call the delivery service with the SAML response. The delivery service also

performs the purpose authorisation request and gets a PAT from the visitor IDaaS. The PAT contains the key attributes “delivery”, “contact”, and “current”. With the given keys, the delivery service can decrypt the user data for the purpose “delivery” as shown in Figure 47. Here, the “addresses” is available for shipping the product, but the “dob” is in ciphertext format.

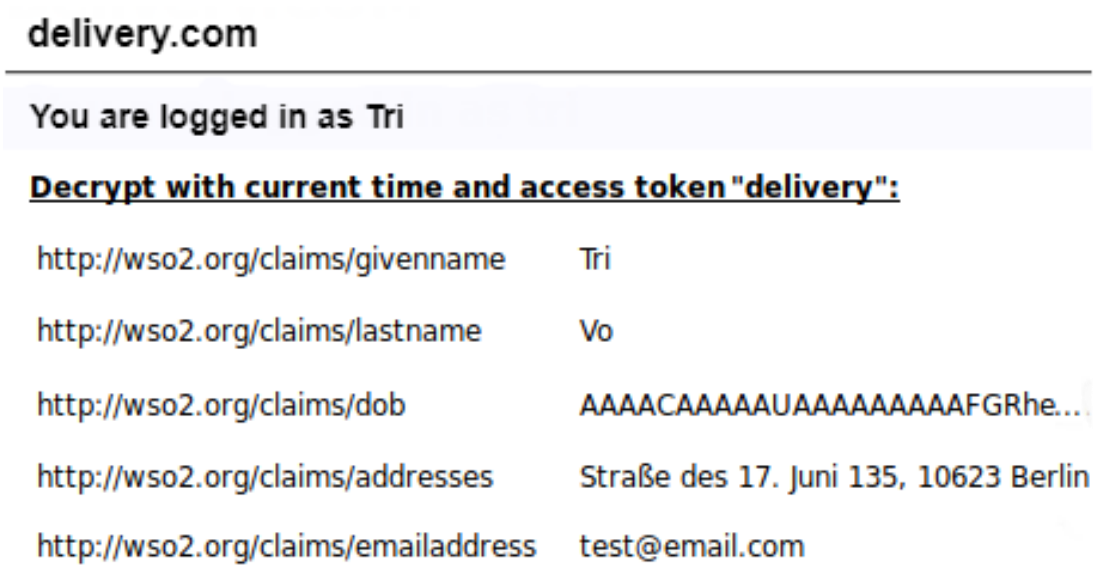

Figure 47: The delivery service GUI

6. **Expired authentication**: When the shopping service submits an expired SAML session, the PAT does not contain the key attribute “current”. Thus, the shopping service cannot decrypt the “dob”. It means after the shopping service completed the current transaction, it cannot reuse the SAML to access the “dob” (even though it was authorised to access the “dob” before).
7. **Expired ciphertext**: When the user authenticates to the home IdP after 15 days, the home IdP issues a TAT with a new access time. Recall that the “addresses” is encrypted with the policy limited to 14 days. This time, the delivery service cannot decrypt the ciphertext “addresses” (see Figure 48).

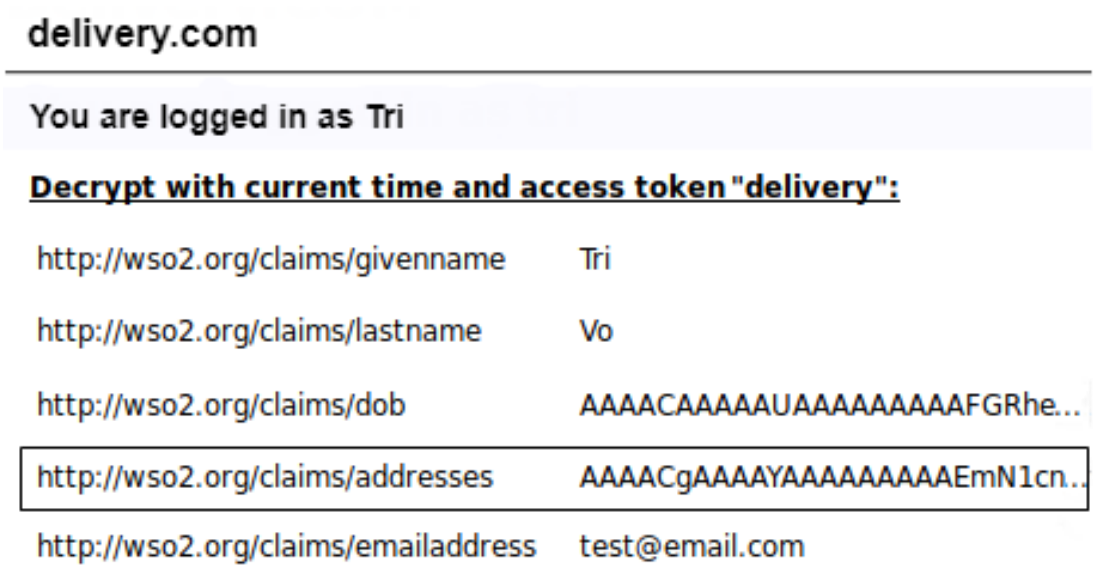

Figure 48: Expired ciphertext "addresses"

In this scenario, the ciphertext is expired, the IdP has fulfilled the intended purpose for collecting the "addresses" (i.e., to deliver the parcel within 14 days). After 14 days, the user may re-encrypt and disseminate his "addresses", if there is a new intended purpose.

8.1.3 PBE Implementation

8.1.3.1 ABE Encryption

Researchers have implemented the ABE scheme [115] by using the jPBC [149], a Java API that wraps the pairing-based cryptosystems written in C [150]. The ABE scheme requires an access policy in the form of a *Linear Secret Sharing Scheme* (LSSS), but the disclosure policies need a mixed formula of Boolean expressions (i.e., AND, OR), simple thresholds (e.g., 1 OF ("marketing", "purchase")), and numerical range (e.g., dayaccess IN (1, 14)). Therefore, researchers first convert the policy string to a Boolean formula and then to an LSSS. To convert a numerical range to a Boolean formula, they borrowed the idea of a segment tree in [151] (see Appendix A2). To convert a Boolean formula to an LSSS, they followed the algorithm of Lewko in [114] (see Appendix A1). In short, an LSSS is a matrix, whereby each row of the matrix

corresponds to a key attribute of the disclosure policies. If one entity can collect enough key attributes that satisfy the disclosure policies, it has enough secret shares to reconstruct the decryption key from the matrix and decrypt the ciphertext (See Appendix A4). This is why an SP has to collect both the TAT and the PAT to access user identities.

8.1.3.2 Ciphertext serialisation

So far, researchers have described that they use ABE for encryption. Actually, they do not use ABE to encrypt PII directly but combine it with the *Advanced Encryption Standard* (AES) [152]. They borrowed this idea from [153] due to the following reasons. AES can encrypt arbitrarily large data without affecting the performance. Furthermore, the ABE scheme indeed encrypts a random element in a Group. If researchers use ABE to encrypt a message (in string format), they may hash the message to the Group element for encryption. However, after decryption, they cannot retrieve the message back from the Group element.

Figure 49 shows the workflow of this hybrid encryption. In the first step, researchers generate an ephemeral key and use it as a symmetric key in AES to encrypt the PII. In the second step, they use ABE to encrypt the ephemeral key. The ciphertext (CT) includes two parts: a header and a payload. The header is the ciphertext output of the ABE scheme (in Section 7.2.2), which contains the attribute index and the encrypted ephemeral key. The payload is the ciphertext output of the AES scheme (i.e., the encrypted PII).

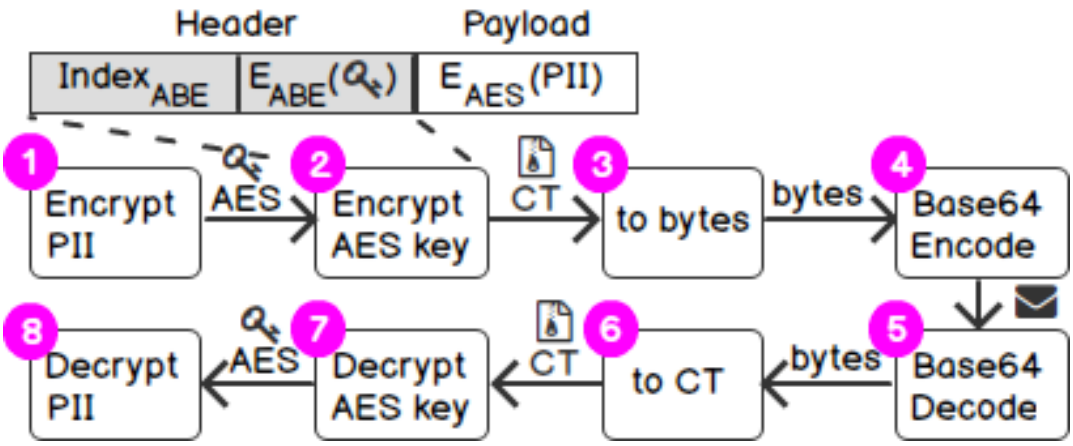

Figure 49: Ciphertext serialisation

For embedding the ciphertext in a SOAP message as well as in an HTTP POST, researchers serialise the ciphertext CT to bytes and then Base64-encode it to a token in a string format (step 3 and 4). The decryption process goes in the reserved order. Researchers decode the token string to bytes (step 5) and then de-serialise bytes to a ciphertext CT (step 6). The jPBC library helps them to implement the serialisation and de-serialisation of the ciphertext. Finally, they decrypt the ciphertext header to get the ephemeral key (step 7) and then use the ephemeral key to decrypt the payload (step 8).

8.1.4 Evaluation of PBE

In the following section, researchers optimise and evaluate the performance as well as the benefits and limitations of PBE.

8.1.4.1 Performance optimisation

Performance is one of the major reasons why user attributes are distributed to federated domains. If the key generation and the decryption process take a long time (e.g., 5 seconds), users will experience a delay in completing a transaction. If the delay were not acceptable, it may be better to keep the PII safe in the central trusted host and not distribute them. Therefore, the solution concept also

focuses on performance. The performance depends on the security levels and the numbers of key attributes in the ABE scheme as follows.

Security levels

An ABE scheme usually computes bilinear pairing operations denoted as $G_1 \times G_2 \rightarrow G_T$, whereby G_1 and G_2 are subgroups of points on an Elliptic Curve (EC) and G_T is a subgroup of a multiplicative group of some extension field [154] (see Appendix A3). The security of an ABE scheme relies on the Discrete Logarithm Problem (DLP) [155]. In particular, it must be computationally infeasible to solve the DLP in three groups G_1, G_2, and G_T. In other words, the size of G_1, G_2, and G_T determines the difficulty of the problem [155]. NIST has recommended the sizes for a secure settings of parings and their validity period as in Listing 13 [156].

Security levels (bits)	EC group size (bits)	Extension field key size (bits)	Period of protection
80	160	1024	disallowed
112	224	2048	2011 - 2030
128	256	3072	> 2030
192	384	7680	> 2030
256	512	15360	> 2030

Listing 13: NIST recommended key sizes in bits

In Listing 13, the first column shows the security levels *k* (i.e., an attacker has to try all 2^k possible keys by brute force). This is also equivalent to the key size of AES. The second column shows the recommended size of an EC group to provide the *k*-bit security. Researchers notice that this size is at least double the key size of AES. The third column shows the extension field key size, which is at least equivalent to the asymmetric encryption RSA [157]. The implementation follows these key size settings of NIST for both AES and ABE.

Key attributes

The decryption of the ABE scheme has *(2l + 1)* pairing operations for *l* key attribute, and the pairing is an expensive operation. Therefore, one way to improve performance is to reduce the number of key attributes. In the beginning, researchers expressed the time policy in a numerical comparison [102]. For example, they used the time policy "notbefore>=1500455744 AND notafter<=1501060500" to present the range of UNIX time from 19.07.2017 to 26.07.2017 UTC. However, this approach takes at most n-bit key attributes for one numerical comparison [102]. Since the time policy had two numerical comparisons, it took 2n-bits key attributes. If a UNIX time format has 32 bits, the time policy took at most 64 key attributes in total.

Finally, researchers have optimised the key attributes using a segment tree [151]. Briefly, each leaf node in the segment tree is a number in the range, and they present a range of numbers by the most common parent of the leaf nodes. With this approach, researchers can present any numerical ranges of size *N* by a set of size *log2(N)* key attributes. Alternatively, researchers can replace the Unix time format with the day policy: "year2018 AND dayaccess IN (270, 256)", whereby "256" and "270" are days in the year 2018. This day policy takes at most 10 key attributes in total, and thus the performance is much faster. The "day" unit is not as precise as "millisecond". However, most examples of data retention in the P3P [17] are in days, so the "day" unit is reasonable.

8.1.4.2 Performance evaluation

Researchers have tested the performance with various data sizes (i.e., 1 byte, 1MB, 10MB, and 100MB) and with various security levels (i.e., 80, 112, 128, 192, and 256 bits). Figure 50 shows the performance result on a CPU i7-8650U, 1.90GHz, 16GB RAM.

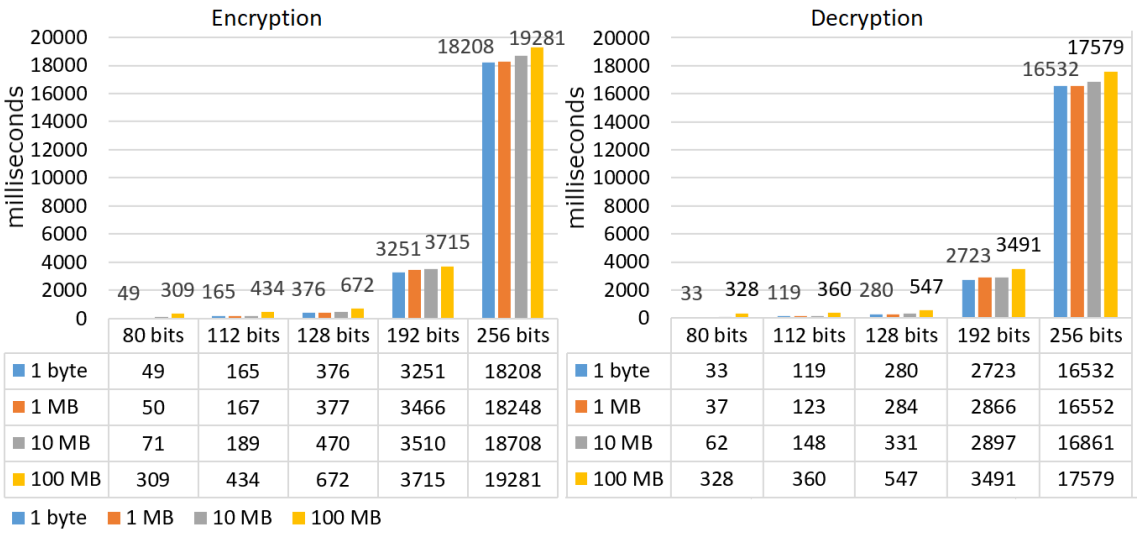

Figure 50: Performance of encryption, decryption in milliseconds (y-axis)

Firstly, researchers look at the performance at the same security level. They notice that various data sizes (i.e., 1 byte, 1MB, 10MB) affect the encryption and decryption time minimally. This is because they use AES to encrypt and decrypt the data. Their AES implementation (without hardware acceleration) achieves a performance of 45MB per second for 128 security bits. As a result, only the encryption and decryption of 100MB data is slightly higher than the other ones.

Secondly, researchers see that the performance is fast for the following security levels: 80, 112, and 128 bits. To give an illusion, for 128 security bits, it took 377ms and 284ms to encrypt and decrypt 1MB, respectively. However, above this level, the performance degrades significantly. For instance, for 192 security bits, it took 10 times longer to encrypt and decrypt the same size. On the other hand, the ABE scheme of Rouselakis and Waters [115] uses the symmetric pairing setting (i.e., $G_1 = G_2$). Several research [155] [158] have experienced that this setting performs fast for security levels below 128 bits, but slow above this level. Similarly, researchers have experienced the same result.

Thirdly, Figure 51 shows the token generation (i.e., the generation of TAT and PAT in total), which is very fast for 80 and 112 security bits (below 415ms). The

performance is also acceptable for 128 security bits (around one second) but slow for 192 security bits (9 seconds). Finally, higher security levels also increase the token size. For 112 and 128 security bits, the tokens add additional overhead to the protocol SAML and OAuth (in total) with 10KB and 15KB, respectively. This token overhead is acceptable.

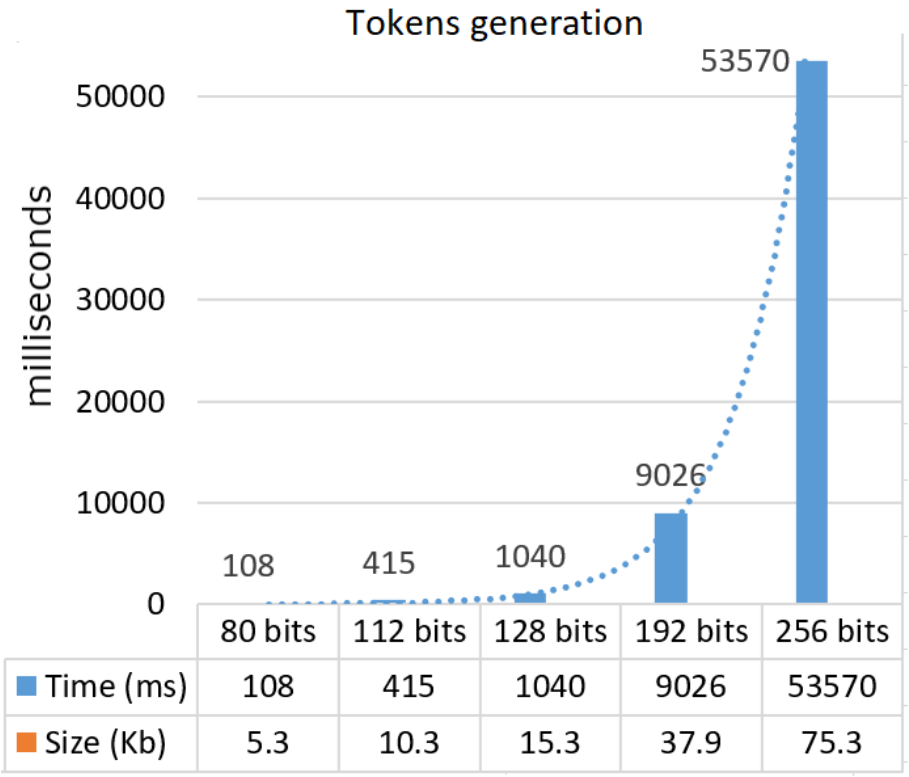

Figure 51: Performance of token generation in milliseconds and token size in KB

In summary, the performance is acceptable for 112 and 128 security bits. However, for 192 bits, service consumers will experience a delay in their transactions (around 12 seconds in total). For 256 bits, the performance is not acceptable.

8.1.4.3 Benefits and limitations

This section discusses whether the PBE approach can solve the issues identified in the introduction:

1. **Protection over an unintended channel**: In PBE, users do not need to interact with each intermediate service in the call chain explicitly. As a result, PBE

prevents identity theft in case of users mistakenly decide, which attributes to disclose to an SP over the frontend. Instead, each intermediate service in the call chain uses the SAML response of the user (as a Single-Sign-On) to perform a purpose authorisation request without the user interaction. The disclosure of the information is limited to the minimum necessary to accomplish the intended purpose in a given time.

2. An honest-but-curious SP: In this case, an SP follows the protocol but tries to learn more information about the PII by altering the SAML response or the tokens. Because the home IdP issues the SAML response with its private key, the SP cannot fake the SAML response to request the visitor IdP for more capabilities than it is allowed. Furthermore, the SP cannot combine different tokens from different transactions or users to gain more capabilities.
3. An honest-but-curious IdP: In this case, an IdP follows the protocol but tries to learn information about the PII. The home IdP can only issue a TAT for a user but does not have the PAT to decrypt the ciphertext. The visitor IdP can only issue a PAT for an SP but does not have the TAT to decrypt the ciphertext. However, this solution concept cannot prevent the collusion attacks between the IdP and its SPs. Here researchers assume that adversaries cannot control both systems at the same time.
4. A dishonest IdP or SP: A dishonest entity is an entity that does not follow the protocol. For example, an IdP authorises a wrong purpose token (with more capabilities) to an SP or an SP forwards its purpose token to another party. In such cases, an honest entity can prove that the other side has violated the protocol and the dishonest entity must pay for the penalty (according to GDPR, an entity may pay a penalty up to 20 million Euro). In the practical example of the Facebook incident, Facebook would prove that it did not authorise the application of Cambridge Analytica for the purpose “analysis”,

so the dishonest application, which forwarded the user data to Cambridge Analytica, would pay for the penalty. It is also worth mentioning that an application can forward PII (in plaintext) to another party after decrypting it. In such cases, researchers say that *the genie is out of the bottle*, and there are no mechanisms to protect the plaintext data.

5. **Insider attack on an IdP**: In this case, an administrator of one IdP may try to steal PII. The administrator may have access to the ciphertext but he does not have enough tokens to decrypt it. However, this approach cannot prevent the collusion attacks between the home and visitor IdP. Here researchers assume that adversaries cannot control both IdPs at the same time.
6. **Malicious host**: In this case, an IdP or an SP deploys a malicious software on the server. Since the host has full control over its execution and fully understands the program, it may change the executions to bypass the control policies [26]. In such a case, if the host changes the cryptographic computing in anyways (e.g., the LSSS matrix), the decryption simply fails.
7. **Chosen-plaintext attack**: The ABE scheme under test is proved to be secured against a non-adaptive chosen-plaintext attack (i.e., an attacker can choose the plaintexts and request the ciphertexts and secret keys in one batch after seeing the global parameters) [115]. To be fully secured, researchers consider an adaptive multi-authority ABE scheme as future work.

8.2 Security infrastructure adaptation

The following sections describe the experiments and evaluation of the security infrastructure adaptation. First, Section 8.2.1 describes the test environment. Section 8.2.2 then explains the experimental results. Section 8.2.3 and 8.2.4

describe the implementation of the security components. Finally, Section 8.2.5 evaluates the implementation.

8.2.1 Test environment

Researchers prove the concept of security topology in a real deployment: Two Web services hosted on two cloud providers establish trust with each other on demand. To prove the portability of this concept, the Web services migrate from a localhost to a cloud provider and then between two cloud providers. To prove the flexibility of the trust model, the experiments have two scenarios: Web services belong to the same and to different organisations.

Figure 52 illustrates two security domains with two cloud providers: OpenStack [53] and AmazonWS [159]. In each domain, researchers have set up a Virtual Machine (VM) IDaaS with an orchestration engine and an IdP.

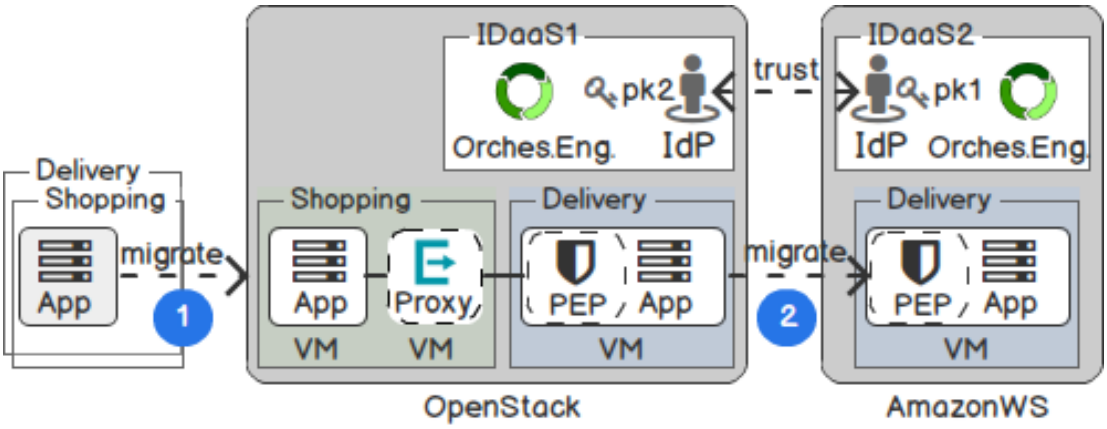

Figure 52: Test environment

Researchers have developed two Web services to simulate a B2B relationship: A shopping service receives purchase orders from the user and then calls a delivery service for shipping the parcel. In the first scenario, the delivery service is a logistic department from the same company. From the technical point of view, they are two application components belonging to the same tenant, and the delivery service trusts the shopping service for user request authentication.

In the second scenario, the delivery service is an external cloud service from a different tenant. Thus, it needs to authenticate and authorise the user request again.

The migration process happens in the following phases. In the first phase, Web services are developed offline. They are unprotected and migrated to OpenStack. The IDaaS in OpenStack will provision and update the AAI implementation for the Web services to trust each other. In the second phase, the delivery service migrates from OpenStack to AmazonWS. In the second domain, the delivery service not only provides SSO access for existing users in the IDaaS of AmazonWS but also maintain the existing trust with the shopping service in OpenStack.

Researchers use the open-source software WSO2 [121] and Alien4Cloud [160] to implement the IdP and the orchestrator, respectively. They have extended the WSO2 IdP for identity federation (i.e., the IDaaS in one domain can authenticate user request with a SAML token issued by the IDaaS in the other domain). Moreover, each Web service runs on a VM. Alien4Cloud controls the VMs and the life cycle of the AAI components inside the VMs. Researchers have developed the implementation artifacts and deployment artifacts for these components. At runtime, the orchestrator sends the artifacts to the VM (via SSH) and calls these artifacts during the life cycle of the components. Alternatively, Alien4Cloud can work with Cloudify [161] to deploy an orchestration agent on the VM (e.g., via Cloud-Init [162]). The orchestration agent then receives the artifacts via a messaging queue in a private network and executes them on the VM [161].

8.2.2 Experimental results

The following section describes the life cycle of the shopping and the delivery service: from the development phase to the migration phase between two cloud providers.

1. **Web service development**: Researchers demonstrate an example workflow for securing a Java Web service with RBAC. In the development phase, developers have no knowledge of an AAI implementation that an IDaaS offers. However, they may define a set of roles for their application as well as which roles are allowed to access which resources. Listing 14 shows the deployment descriptor of the delivery service. In this example, developers allow the role “admin” (line 4) to access a Web resource.

```
1 <security-constraint>
2   ...
3   <auth-constraint>
4     <role-name>admin</role-name>
5   </auth-constraint>
6 </security-constraint>
```

Listing 14: An example of the deployment descriptor of the Web application

Here researchers want to demonstrate that only developers understand their code (or configurations) and know which local roles they implemented. If developers implement ABAC for their applications, the developers may not need to define any roles.

2. **Topology description**: In this step, a security architect describes the security topology. Figure 53 shows a Graphical User Interface (GUI) that helps the architect to describe the topology for the delivery service. The GUI is the presentation of the topology model. On the left side, the architect defines a “WebService” node running on a “Glassfish” application server of a “Compute” VM. He also wires an “InterceptingWebAgent” node to the

application endpoint capability of the “WebService” node to define a “protect” relationship between them.

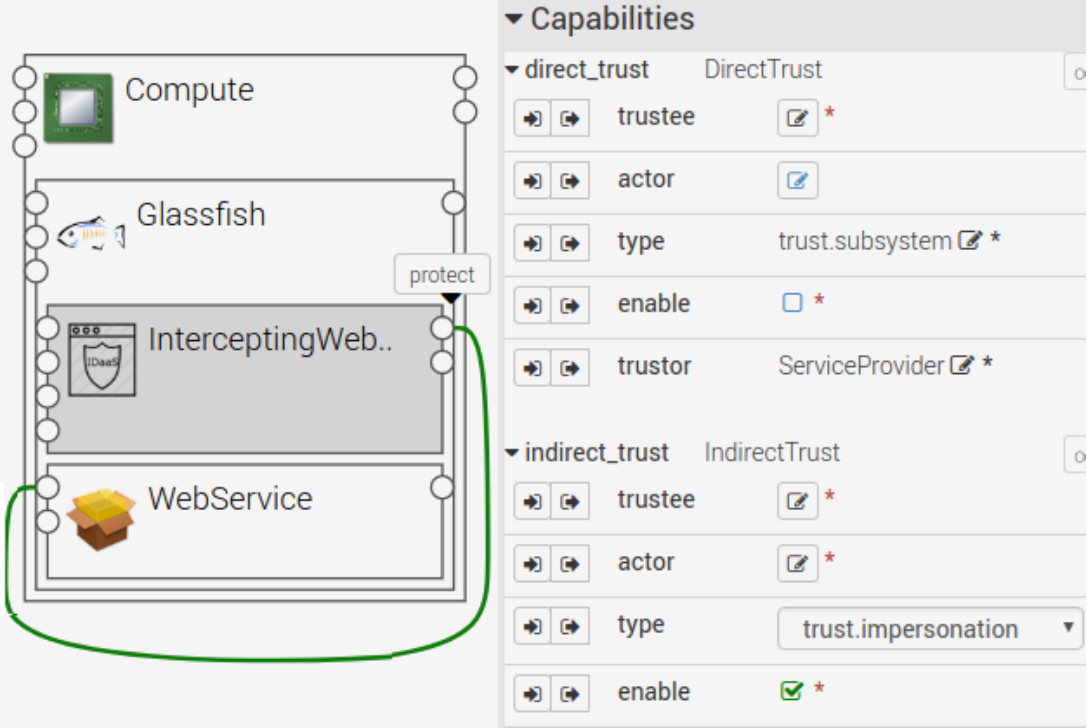

Figure 53: The topology of the delivery service (GUI)

On the right side, he can specify the trust capabilities of the “InterceptingWebAgent”. In the first scenario, the delivery service is from the same organisation, so he chooses the “direct_trust” capability and specifies the public key of the shopping service. In this trust model, the shopping service is the “actor”, and the architect can specify a role (e.g., admin) for the shopping service in the “actor” property.

In the second scenario, the service consumer is still unknown, so the architect chooses the “indirect_trust” capability. In this trust model, the user is the “actor”, and the delivery service has to authorise the user based on his identity. Here the architect can specify the endpoint of IDaaS1 as the “trustee” to issue user identity. He may allow the service consumer to act on behalf of the user, so he selects the value “trust.impersonation” in the “type” property (optionally, he may choose “trust.delegation”). Listing 15 shows the

generated topology description. The nodes have been instantiated with the trusted endpoint of IDaaS1 (line 16). The architect also specifies that he needs an “email address” and a user “group” from IDaaS1 (line 18 - 20) for his application.

```
1  InterceptingWebAgent:
2    type: idaas.nodes.InterceptingWebAgent
3    requirements:
4      - protect:
5          node: WebService
6          capability: tosca.capabilities.Endpoint
7          relationship: idaas.relationships.WebagentProtectWs
8      - host:
9          node: Glassfish
10         capability: idaas.capabilities.WarContainer
11         relationship: idaas.relationships.WebagentHostedOnGf
12     capabilities:
13       indirect_trust:
14         properties:
15           trustee:
16             sts_wsdl:"https://80.158.19.63:9443/wso2carbon-sts?wsdl"
17           actor:
18             claims_mapping:
19               wso2.org/claims/emailaddress: "HTTP_EMAIL"
20             group_mapping: "http://wso2.org/claims/groups"
21           type: "trust.impersonation"
22           enable: true
23 WebService:
24   type: idaas.nodes.WebService
```

Listing 15: The topology description of the delivery service (indirect trust)

In the topology of the shopping service (Figure 54), the architect uses a “Proxy” node to intercept outbound requests from the shopping service and connects the “Proxy” node to a “PEP” node.

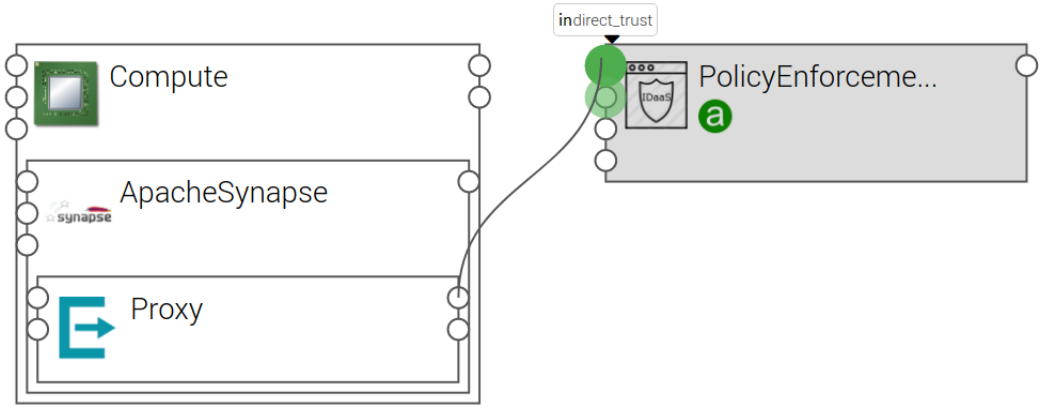

Figure 54: The topology of the shopping service (GUI)

Listing 16 shows the generated topology description of the shopping service. Recall that the PEP node is an abstract node that represents a partner service. The architect may use a TOSCA "node_filter" to define his trust criteria for selecting a partner service (line 9). If the partner service is known beforehand (e.g., the delivery service), he chooses the "direct_trust" capability and specifies the public key of the delivery service. Otherwise, he chooses the "indirect_trust" capability and specifies IDaaS1 as one of the trusted endpoints of the Proxy (line 14).

```
1  Proxy:
2    type: idaas.nodes.Proxy
3    requirements:
4      - service_provider: service1
5
6  # a placeholder of service provider
7  service1:
8    type: idaas.nodes.PolicyEnforcementPoint
9    node_filter:
10     capabilities:
11       - indirect_trust:
12           properties:
13             trustee:
14               sts_wsdl: {valid_values:["https://80...."]}
```

Listing 16: The topology description of the shopping service (indirect trust)

3. **Provisioning**: In this step, a tenant administrator provisions the delivery service to OpenStack. The orchestrator uses the OpenStack APIs (and the credential of the administrator) to boot a VM, deploys the Web service, and calls the interface "pre_configure_source" of the "InterceptingWebAgent" to protect it. For direct trust, the public key of the shopping service is imported to the truststore of the application server. For indirect trust, the Web agent is configured to trust IDaaS1 and intercept client requests on the message layer (see the Web agent implementation). Also, the WS-SecurityPolicy is generated according to the trust model. Listing 17 shows an example of the generated security policies that conform to the indirect trust model.

```
1  <sp:TransportBinding>
2   ...
3  </sp:TransportBinding>
4  <sp:SupportingTokens>
5   <wsp:Policy>
6    <sp:IssuedToken sp:IncludeToken=".../AlwaysToRecipient">
7     <sp:Issuer>
8      <Address>https://80.158.19.63:9443/services/wso2carbon-sts?wsdl</Address>
9     </sp:Issuer>
10    <sp:RequestSecurityTokenTemplate>
11     <t:TokenType>urn:oasis:names:tc:SAML:2.0:assertion</t:TokenType>
12     <t:KeyType>http://schemas.xmlsoap.org/ws/2005/02/trust/Bearer</t:KeyType>
13     <t:KeySize>256</t:KeySize>
14     <t:Claims>
15      <ic:ClaimType Uri="http://wso2.org/claims/groups"/>
16      <ic:ClaimType Uri="http://wso2.org/claims/emailaddress"/>
17     </t:Claims>
18    </sp:RequestSecurityTokenTemplate>
19   </sp:IssuedToken>
20  </wsp:Policy>
21 </sp:SupportingTokens>
22 <ramp:RampartConfig xmlns:ramp="http://ws.apache.org/rampart/policy">
23  ...
```

Listing 17: An example of the generated WS-SecurityPolicy for indirect trust

In this example, the WS-SecurityPolicy also contains the platform-specific implementation of the Web agent (e.g., the Apache Rampart configs of the truststore and keystore on lines 22 to 23).

4. **Integration test of the Web agent**: To validate the deployment, the orchestrator calls the interface “add_target” of the “InterceptingWebAgent” to run an integration test. The test sends a dummy SOAP request (with no authentication) to the Web agent endpoint. If the request is denied, the Web service is secured successfully. Otherwise, the orchestrator disables the delivery service from the application server. If the test is successful, the orchestrator publishes the endpoint as well as the trust capability of the delivery service (from Listing 15) in its registry.
5. **Update**: In this step, the shopping service signs a business contract to use the delivery service. An administrator provisions the shopping service to OpenStack. Before the deployment, the orchestrator allows the administrator to confirm the security matching between the two services. Then the orchestrator sets up the Proxy node according to the trust capability of the

delivery service (Figure 55). For direct trust, it imports the public key of the delivery service to the truststore.

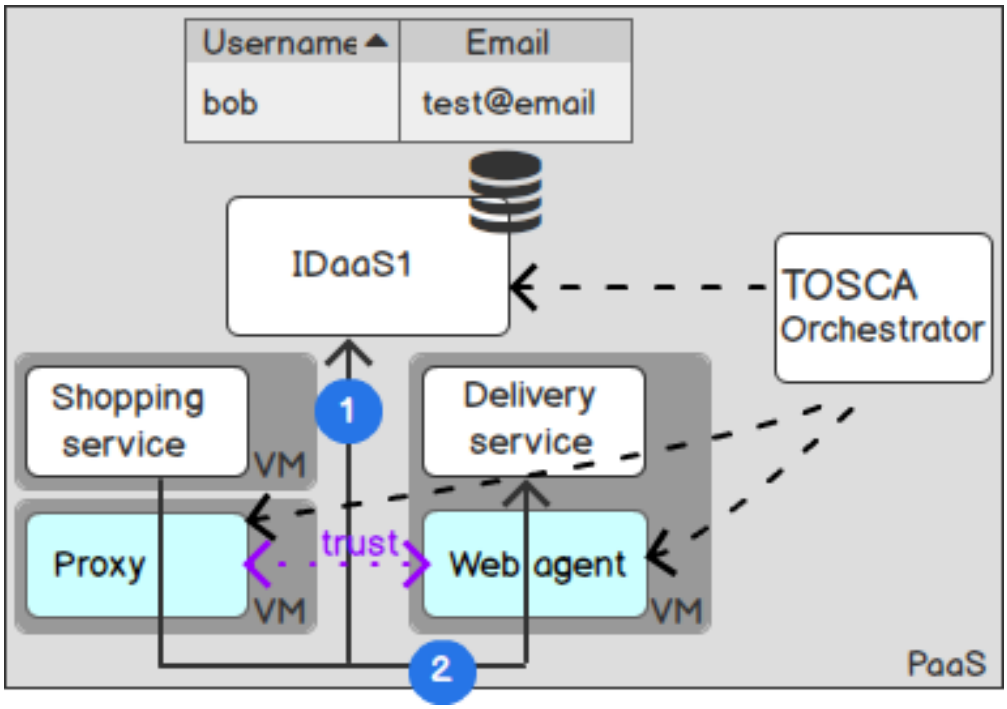

Figure 55: TOSCA Orchestrator adapts indirect trust relationship between services

For indirect trust, it configures the Proxy to trust IDaaS1 and to impersonate the user. In particular, the orchestrator sees that the delivery service requires an "email address" (as specified in the "claims_mapping"). Thus, it configures the outgoing Proxy of the shopping service to include an email address in the request (step 2). If the shopping service does not have this attribute in its topology description (i.e., the shopping service neither requires this attribute for running the service nor currently has it in the application's context), the orchestrator configures the Proxy to ask IDaaS1 for this user attribute (step 1).

6. **Integration test of the Proxy**: The orchestrator also calls the interface "add_target" of the Proxy node to check if the delivery service's endpoint is accessible. According to the trust capability, the Proxy can call the delivery service directly (i.e., direct trust) or indirectly with the token issued by IDaaS1 (i.e., indirect trust). Otherwise, the test will fail.

7. **Termination**: In the second test phase, the delivery service migrates from OpenStack to AmazonWS. In the domain of OpenStack, the orchestrator removes any credentials of the delivery service from the Proxy node, and then it terminates the delivery service VM.
8. **Migration**: In this step, the orchestrator in each domain updates the deployment of the Web service under its control (Figure 56).

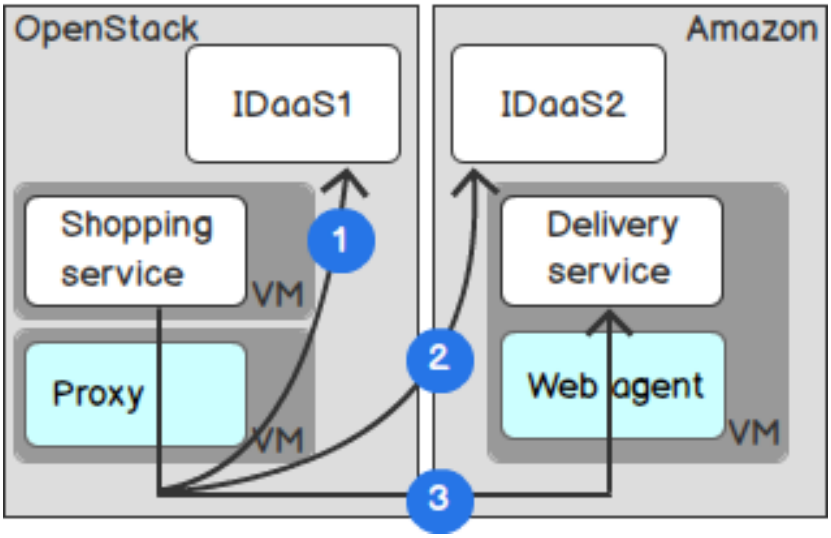

Figure 56: Indirect trust is established again after the migration

As depicted in Figure 56, in the AmazonWS domain, the topology of the delivery service is updated with the new trusted endpoint (i.e., IDaaS2). Then the orchestrator deploys the delivery service and configures the "InterceptingWebAgent" to trust IDaaS2. In the domain of OpenStack, the delivery service is now an external service. The administrator defines a PEP node (as a substitution for the external service) with the "trustee" updated to the endpoint of IDaaS2. Then the orchestrator can read this trust capability and configures the "Proxy" node for identity federation (see the Proxy implementation).

8.2.3 Proxy implementation

Researchers use the open-source software Apache Synapse [163] to implement the Proxy that mediates outbound requests from a Web service client. For direct trust, the Proxy acts as a gateway to sign and encrypt the message on the transport layer. The implementation of direct trust is straightforward by importing the certificate of the Web service to the truststore of Synapse.

For indirect trust, researchers follow the design pattern of identity proxy in [118]. In this design pattern, the STS Client directly interacts with two STS and the Web service sequentially from left to right (see Figure 57). The STS Client first authenticates to STS1 to receive a SAML assertion about a user (step 2) and then submits the SAML assertion to the second one (step 3). The second STS acts as a proxy gateway, which validates the received assertion and transforms it for the STS Client to access the Web service (step 4). In this case, the Web service only trusts STS2 as the proxy gateway for issuing assertions.

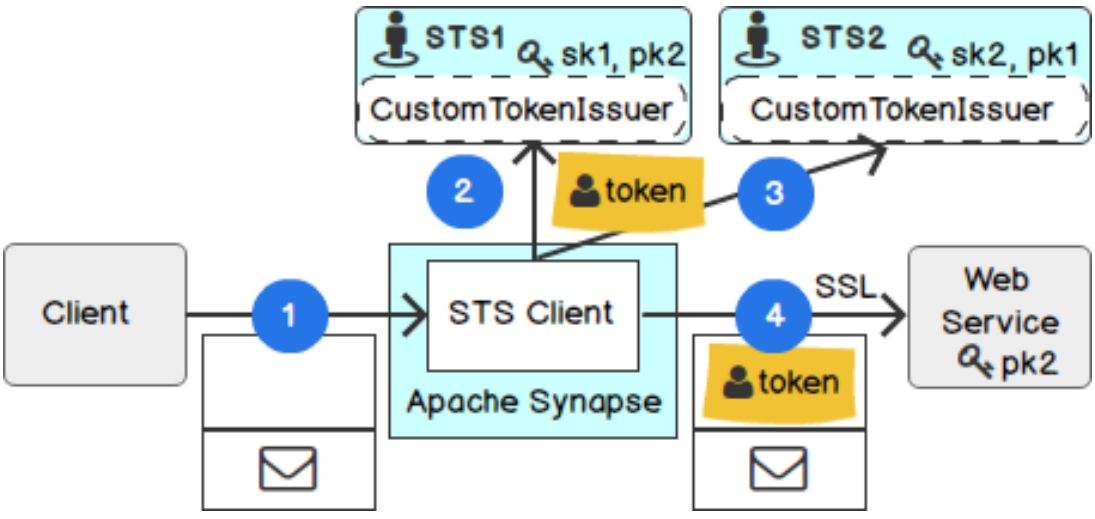

Figure 57: Outbound Proxy implementation

8.2.4 Intercepting Web agent implementation

In Figure 58, the Web agent receives the SAML assertion from the STS Client (step 1) and propagates the required claims to the Web service (step 2).

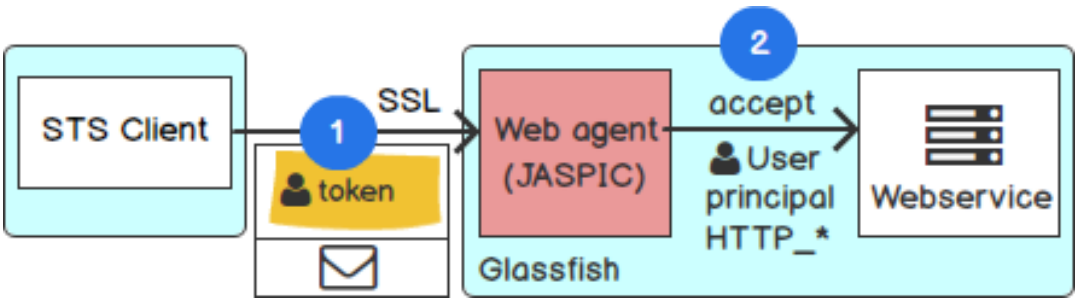

Figure 58: Intercepting Web agent implementation

The Web agent loads the public key of the STS server, which the orchestrator imported to Glassfish previously, and uses it to validate the signature of the SAML assertion. Afterwards, it authenticates the user subject in the SAML to the container-managed authentication.

8.2.5 Evaluation of security infrastructure adaptation

Cost-cutting

In the experiment, developers are not required to understand any protocols (e.g., SAML, WS-Trust), to relearn any configurations (e.g., the domain descriptor of the application server), any proprietary APIs of an application server (e.g., the JAAS implementation of Glassfish), or an IdP (e.g., WSO2). Instead, it is assumed that developers do not know the AAI implementation that protects their applications at runtime. In the experiment, developers only defined some local roles for the Web service (using the standard application descriptor web.xml) but no other security implementations or configurations. Thus, the approach reduces the cost for analysts and security experts to adapt AAI for an application in a cloud environment.

Portability

The implementation of the Web agent faces two obstacles of interoperability and portability. Portability is the ability to move an application from one cloud provider to another one at the lowest possible cost, effort, and time [25]. The

TOSCA specification achieves the portability of cloud applications by using *declarative processing* and *imperative processing* [164]. Declarative processing requires a precise definition of the semantics of node types and relationship types so that an orchestrator can interpret these types and deploy them in a target cloud provider (i.e., the orchestrator has already supported these types and implemented APIs of multiple cloud providers to deploy these types). Because the security nodes inherit from the normative node types of TOSCA (e.g., "WebApplication"), the orchestrator can interpret the nodes as a Web application for the deployment. By inheriting the normative relationship types (e.g., "ConnectsTo" and "HostedOn"), the orchestrator can control the dependency and the life cycle of the security components. On the other hand, imperative processing requires explicit instructions on how to manage the components. In the implementation, researchers provide deployment artifacts (a Jar archive) and implementation artifacts (shell scripts) for the PEP and the Proxy. These artifacts are examples of imperative processing. Another IDM vendor may replace these artifacts by theirs if they have a specific solution.

In addition, the cloud architect may host his Web service on an arbitrary application server. Therefore, the Web agent needs to be portable on various application servers as well. Researchers follow the relative new specification of Java Authentication Service Provider Interface for Containers (JASPIC) [165] to implement the Web agent. By using JASPIC, it is not necessary to use proprietary codes of Glassfish for container-authenticated management.

Interoperability

Interoperability is the ability of two or more components to exchange information and to use the exchanged information [25]. In the experiment, the Web agent needs to exchange the user identity with the application in the backend, but the

backend is a black box. Furthermore, developers implement Web services without the knowledge of the AAI implementation. In the experiment, the security architect supports the interoperability between the Web agent and the application in the backend by providing the "group_mapping" and "claims_mapping" in the trust capability previously. By reading the "group_mapping", the Web agent understands which claim is mapped to the group principal (e.g., http://wso2.org/claims/groups). By reading the "claims_mapping", the Web agent understands which HTTP headers to propagate to the backend (e.g., HTTP_EMAIL).

It is worth mentioning that, if researchers implement the Web agent for scripting languages (e.g., PHP or Ruby), they will use HTTP headers, because identity propagation in HTTP headers is considered as programming and platform-independent for any applications to obtain information about a request [136]. If they implement the Web agent for Microsoft .NET applications, they will use the Windows Identity Foundation (WIF) [142]. By using WIF, researchers will not face the obstacles above, because WIF has standardised its identity model for propagating a SAML assertion (see Section 6.3.1.4). In this thesis, researchers test a Java application, because the Web agent implementation in Java has such obstacles for security adaptation to consider.

8.3 Conclusion

This chapter has evaluated the lifecycle of the PII in the testbed. During the lifecycle, the PII is completely encrypted. However, an intermediate SP (e.g., the shopping and the delivery service) can decrypt the ciphertext to fulfil certain purposes. The experiment also evaluates the cases when the ciphertext is expired or when users do not limit the access time for the current transaction. For the first case, when the ciphertext is expired, the visitor IDaaS as the data

controller has fulfilled its purposes to collect PII, and SPs as the data processor has fulfilled its purposes to process the data and are not authorised to access it further. For the second case, the user authentication is expired, but SPs can still access PII if the users allow them to do so (e.g., for marketing or contact purpose). Finally, this reference architecture can identify a dishonest party that does not follow the protocol. However, it cannot prevent the collusion attacks between the home and visitor IDaaS.

This chapter has evaluated the adaptation of the required AAI implementation for two business scenarios and in two cloud providers. When an SP (e.g., the delivery service) migrates from OpenStack to AmazonWS, IDaaS can adapt the required AAI implementation to maintain the trust between the two SPs in the new environment. With this, the portability issue has been solved. Furthermore, whether the SPs belong to the same or to different organisations, IDaaS can adapt the AAI implementation to propagate the required PII from the original caller via the intermediate SP (i.e., the shopping service) to the end SP (i.e., the delivery service). With this, the interoperability issue has been solved. Also, IDaaS provisions the security components (e.g., the intercepting web agent) with the functionality of PBE to decrypt the ciphertext for the application component in the backend to use. This experiment proved that the application developers do not need to have special knowledge about the underlying security protocols (or any new protocols such as PBE).

Chapter 9

Summary and future work

This chapter summarises the research. Section 9.1 and Section 9.2 describe the achievements and limitations of the research, respectively. Finally, Section 9.3 describes the opportunities for further research.

9.1 Achievements

In e-commerce, SPs require flexible and secure access control mechanisms for identity federation, but this security feature is not a core competency. In the cloud environment, SPs may need to adapt their AAI implementations and propagate PII of the original caller on demand. Thus, they may prefer outsourcing their IDM to a trusted third party that strengthens their security implementation and may reduce their cost. On the other hand, users access multiple cloud services in federated domains but may want to protect their privacy.

9.1.1.1 Related work

Prior research has not proposed any solutions to adapt the AAI implementations for cloud services in multiple cloud providers (i.e., the portability issue). None of them can control the identity propagation from the original caller to the end SP to complete a business transaction on demand (i.e., the interoperability issue). Furthermore, they focus on protecting the application components against possible threats but do not protect PII across intermediaries in the call chain.

On the other hand, many efforts in the past 10 years have been taken in protecting PII. These approaches target certain issues but still have limitations. For instance, users require interacting with the SPs over the frontend; they neither protect identity propagation between intermediaries nor against an untrusted host. It is worth mentioning that human-PC link is the weakest link

compared to the links between the PC, SP, and IdP. Therefore, the human link is the major target of identity theft.

9.1.1.2 The approach of using IDaaS

Due to the missing gap above, this thesis proposed:

IDaaS is a trusted Identity and Access Management that (1) adapts the authentication and authorisation infrastructures of cloud services to complete a business-to-business transaction on demand and (2) protects the dissemination of PII in federated security domains.

To achieve the first goal, the thesis considers the application components as *unprotected* and IDaaS provisions the AAI implementations to protect them at runtime. Because the implementation of an application is like a black box, IDaaS first needs to understand the topology view of the application. For this reason, the research separated the implementation of the security mechanisms and the identity propagation from the application business logic. It then collected the authorisation design patterns that the developers frequently use to protect their applications and modelled them in a novel *security topology*. By using the security topology, the adaptation process can understand the application (i.e. the network and the hosting compute node, the relationship between the security components and the application components, and the identity mapping between them). Then the adaptation process can provision the AAI implementations, establishes a dynamic trust relationship between cloud services according to the security topology, and evaluates any changes in the runtime environment to conform to the model.

9.1.1.3 Benefits of using security infrastructure adaptation

The adaptation of the security infrastructure gains three benefits:

1. First, the experiments showed that whenever a cloud service (e.g., the delivery service) migrates from one cloud provider (e.g., OpenStack) to another one (e.g., AmazonWS), the complexity of the security infrastructure does not move with the application infrastructure. The adaptation process enforced a constraint security infrastructure for cloud services that is independent of multiple cloud providers. With this result, the research solved the portability issue of security infrastructures in multiple cloud providers.
2. Second, the experiments proved that when the business comes up with a new e-commerce scenario (e.g., a scenario involves one organisation or multiple organisations), IDaaS provisions the security infrastructures to propagate PII from the original caller to the end SP for the given business scenario. With this result, the research solved the interoperability issue between the security infrastructures and SPs in the call chain.
3. Third, the experiments demonstrated that developers do not need to have specialised security knowledge to understand various protocols, to relearn and implement repetitive tasks using proprietary APIs of any vendors.

To achieve the second goal, the thesis proposed a novel approach to *Purpose-based Encryption* (PBE). To the best of the research' knowledge, this is the first approach to combine *Purpose-based Access Control* and *Attribute-based Encryption* to protect the confidentiality of disseminated data with multi-authorities support. In particular, users encrypt their PII with disclosure policies. A cloud service can decrypt the ciphertext if and only if it receives a time access token and a purpose access token that satisfy the disclosure policies. The research showed that access to PII could be revoked if the ciphertext or the authentication was expired.

9.1.1.4 Benefits of using PBE

The approach of using PBE gains three benefits:

1. First, the experiments proved that PBE protects PII from any intermediary SPs in the call chain (e.g., the shopping and the delivery service). The evaluation also asserted, if an untrusted host changes the execution of plaintext code to bypass the disclosure policies (i.e., the LSSS matrix), the decryption simply fails.
2. Second, the experiments showed that existing IDM systems and applications could easily adapt PBE because PBE integrates with the standard protocols SAML and OAuth.
3. Third, the experiments evaluated that PBE is suitable for sharing sensitive user information in a large distributed and heterogeneous environment (e.g., between two cloud providers). Unlike a traditional access control (e.g., RBAC) that allows a local role to access data, or encryption schemes (e.g., RSA [157] and Identity-based Encryption [82]) that requires an explicit identity to decrypt the data, PBE does not limit access control to a local role or an explicit identity.

9.2 Limitations

The reference architecture of IDaaS separates the roles and responsibilities of the home and visitor IDaaS. As a result, if a dishonest entity does not follow the protocol, an honest entity can prove that the other side has violated the protocol and the dishonest entity must pay for the penalty. However, this reference architecture cannot prevent the collusion attacks between the home and the visitor IDaaS. In such a case, an adversary takes control of both IDaaS and collects the time access token and the purpose access token to decrypt the ciphertext. Finally, the proposed architecture requires organisations in various

security domains federating their IdPs with each other. However, one organisation may refuse to do so.

To demonstrate a complete lifecycle of PII in the testbed, the research prototype implemented a fully functional PBE as an OSGi plugin to the open-source IdP WSO2. It also provided a library for a client application (e.g., to encrypt PII) and for an intercepting web agent (e.g., to decrypt the ciphertext). Although the research prototype was implemented in Java and demonstrated in WSO2, the thesis integrates PBE to the standard protocols SAML and OAuth. Therefore, the implementation is independent of any programming languages and from any IdP vendors, but not limited to WSO2. Also, the evaluation of PBE in the testbed is closed to a production environment with the exception that the IdPs and the SPs have to exchange the public keys of the ABE scheme. The exchange of public keys between entities is not in the scope of the research prototype.

The only limitations in implementing PBE is that it relies on the underlying ABE scheme in terms of performance and its security model. In comparison to previous work, the research has improved the performance of ABE significantly. In particular, the implementation can express the disclosure policies in a numerical range with less impact on the performance. As a result, the performance is fast for the security levels of 112 and 128 bits. However, the ABE scheme of Rouselakis and Waters [115] still has a performance issue on the security level of 256 bits. Regarding the security model, the given ABE scheme is not secured against an adaptive chosen-plaintext attack.

9.3 Directions for future work

The research has the following directions for future work of PBE (Section 9.3.1) and security infrastructure adaptation (Section 9.3.2).

9.3.1 Future work of PBE

Most proposed schemes of ABE in recent years are built on Type A curves [166]. Since the ABE scheme of Rouselakis and Waters [115] still has a performance issue on the security level of 256 bits, future work may research a multi-authority ABE scheme that is built on Type F curves [166], which is known to provide faster performance in the security level of 256 bits [155]. In addition, the research has not found any multi-authority ABE schemes that are proven under an adaptive security model and support a large universe of attributes at the same time, which leaves rooms for future research.

The thesis has advanced the field of IDM for cloud computing. However, a number of areas for future work can be built upon the results achieved, especially in edge computing [167] and Internet of Things as follows. In the Internet of Things, sensor devices may need to collect sensitive information from data producer. These devices need to cooperate with each other for some sensitive jobs without the involvement of the users. It means the machines may talk to each other without user interaction. As a result, users may lose control of their personal data. For example, a cleaning robot stores a map of a user's apartment in the robot's central service, or a mobile device processes user's voices and stores them in a central service in a text format.

In such cases, a user Bob may want to protect his privacy from multiple sensor devices that collect his data intensively without his interactions. One may use PBE as follows. When data producers send data, they automatically encrypt the data with PBE. As a result, only sensor devices from specific domains could process a portion of Bob's data if they have the right purposes and in the given time. For example, in the case of the cleaning robot, users may specify the purposes of using their personal data in the settings of the robot. As a result, the

functionalities of PBE (integrated into the robot) may remove the user location and encrypt the apartment map with the “pseudo-analysis” purpose. It may also encrypt the “email address” with the “marketing” purpose before sending it to the central storage. Then if any services (e.g., the robot central service or any partner companies of the robot provider) want to access this information, they may send a purpose authorisation request to the IDaaS for the purpose of “pseudo-analysis” (i.e., to improve the functions of the robot). If they want to contact Bob for the “marketing” purpose, they may have access to his email address but not his apartment map. Furthermore, these services may require a time access token from the home IDaaS of Bob when he configures the settings of the cleaning robot. In this case, Bob could limit the services to process the map of his apartment in a period (e.g., in a week) but not in a year.

In the example above, PBE does not need to define an explicit sensor device that process Bob’s data in the disclosure policies. Therefore, PBE also protects the confidentiality of Bob’s data from unknown devices and from the central data storage, of which Bob may not be aware. If the sensor device or the central data storage is a malicious host and tries to alter the disclosure policies or the access tokens, the decryption simply fails.

9.3.2 Future work of security infrastructure adaptation

In the security infrastructure adaptation, the orchestrator attests its own deployment using the default integration tests. Future work may improve the certification of the security infrastructure, whereby a trusted third party can attest the deployment using the Trusted Platform Module as in [168].

On the other hand, microservice architecture [169] decomposes an application into a smaller set of focused and independently deployable services.

Microservices may communicate among themselves using lightweight mechanisms such as REST APIs. As a result, the application developers may not have a global view of the entire applications and attackers can exploit this complexity to launch attacks against the applications [52]. The approach of describing the security topology in this thesis may help to protect microservices on demand and removes this burden from the application developers.

In edge computing, the computation is largely performed on the far edge node [167]. For example, an auto-driving car may have a microservice responsible for calculating the direction running on the edge node closest to the auto. The microservices of all cars running on the same street or in the same area may exchange some information about the traffic among themselves. As a result, the microservices may establish trust with their unknown peers frequently and dynamically. In such a case, each edge node may have a visitor IDaaS that adapts the AAI implementations and establishes trust among the microservices automatically. In such a dynamic environment, this approach removes the burden to implement AAI manually to establish trust between microservices.

9.4 Conclusion

In recent years, organisations have federated their IDM systems to reduce the costs of managing users individually. As a result, user information is not stored in one organisation or one location but disseminate to multiple ones. Also, the architecture of edge computing and microservices tend to split a heavy central service to multiple lightweight and distributed ones. As a result, user information is processed by multiple distributed services in various locations and heterogeneous security domains, which makes it even harder for performing access control. For this reason, traditional access control that addresses an

explicit entity (or local roles) to access certain objects or a single authoritative resource that issues an authorisation credential, may not fulfil the new requirements.

The most important contribution of the thesis is to define explicit roles and responsibilities for multi-authorities in heterogeneous security domains: users can choose an authoritative resource as a home domain that issues a (global) time access token for them to log in to multiple services in heterogeneous security domains. This home domain could be a trusted authoritative resource from the government or a banking system. Then in the other security domains, an authoritative resource issues a (local) purpose access token for the relying SPs. This visitor domain could be an organisation, a cloud provider, or a government. The cryptographic combination of both tokens enables access control for distributed services in various locations and heterogeneous security domains.

In the previous examples, the thesis has motivated the use case of a cleaning robot but not limited to it. In general, the automation of the machines in doing human tasks may require automated protection of personal data without human interaction. To tackle these issues, the thesis of IDaaS contributes a solution that is compliant with GDPR to an auto-protection of personal information in widely distributed systems.

References

[1] D. Chadwick, “Federated Identity Management,” in *Foundations of Security Analysis and Design V SE - 3*, vol. 5705, A. Aldini, G. Barthe, and R. Gorrieri, Eds. Springer Berlin Heidelberg, 2009, pp. 96–120.

[2] A. Bücker, F. Werner, H. Hinton, H. P. Hippenstiel, M. Hollin, R. Neucom, S. Weeden, J. Westman, A. Buecker, W. Filip, H. Hinton, H. P. Hippenstiel, M. Hollin, R. Neucom, S. Weeden, J. Westman, A. Bücker, F. Werner, H. Hinton, H. P. Hippenstiel, M. Hollin, R. Neucom, S. Weeden, and J. Westman, *Federated Identity Management and Web Services Security with IBM Tivoli Security Solutions*. Indianapolis: IBM Redbooks, 2005.

[3] C. Schläger, M. Sojer, B. Muschall, and G. Pernul, “Attribute-Based Authentication and Authorisation Infrastructures for E-Commerce Providers,” in *E-Commerce and Web Technologies SE - 14*, vol. 4082, K. Bauknecht, B. Pröll, and H. Werthner, Eds. Springer Berlin Heidelberg, 2006, pp. 132–141.

[4] D. Ferraiolo, D. Kuhn, and R. Chandramouli, “Role Based Access Control.” 15th National Computer Security Conference, pp. 554–563, 2007.

[5] Wikipedia contributors, “Policy-based management,” *Wikipedia, The Free Encyclopedia*, 2016. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Policy-based_management&oldid=700392159. [Accessed: 18-Jan-2016].

[6] “eXtensible Access Control Markup Language (XACML) Version 3.0,” *OASIS Standard*, 2013. [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html. [Accessed: 22-Jan-2013].

[7] R. Wagner, “Identity and Access Management 2020,” *ISSA J.*, 2014.

[8] V. Andrikopoulos, T. Binz, F. Leymann, and S. Strauch, “How to adapt applications for the Cloud environment,” *Computing*, vol. 95, no. 6, pp. 493–535, Dec. 2012.

[9] A. Nanda and M. B. Jones, “Identity selector interoperability profile v1. 5,” *Microsoft Corporation*, 2008. [Online]. Available: http://download.microsoft.com/download/1/1/a/11ac6505-e4c0-.

[10] J. Camenisch and E. Van Herreweghen, “Design and implementation of the idemix anonymous credential system,” *9th ACM Conf. Comput. Commun. Secur. CCS 02*, p. 21, 2002.

[11] D. B. Skillicorn and M. Hussain, “Personas : Beyond Identity Protection by Information Control A Report to the Privacy Commissioner of Canada,” *Identities*, no. April, 2009.

[12] S. Landau and T. Moore, “Economic tussles in federated identity management,” *First Monday*, vol. 17, no. 10, 2012.

[13] “Identity in the Cloud Use Cases Version 1.0,” *OASIS Committee Note 01*, 2012. [Online]. Available: http://docs.oasis-open.org/id-cloud/IDCloud-usecases/v1.0/cn01/IDCloud-usecases-v1.0-cn01.html.

[14] K. Cameron, “The Laws of Identity,” 2005. [Online]. Available: http://www.identityblog.com/?p=352/#lawsofiden_topic3. [Accessed: 13-May-2005].

[15] “General Data Protection Regulation, final version,” *Official Journal of the European Union*, 2016. [Online]. Available: https://eur-lex.europa.eu/. [Accessed: 04-May-2016].

[16] ITU-T, “NGN identity management framework, Recommendation Y.2720,” *Y.2720*, 2009. [Online]. Available: https://www.itu.int/rec/T-REC-Y.2720/. [Accessed: 21-Oct-2009].

[17] W. Rigo and S. Matthias, “The Platform for Privacy Preferences 1.1 (P3P1.1) Specification,” *W3C Recommendation*, 2006. [Online]. Available: http://www.w3.org/TR/P3P11/. [Accessed: 13-Nov-2006].

[18] N. Delessy, E. B. Fernandez, and M. M. Larrondo-Petrie, “A Pattern Language for Identity Management,” *Comput. Glob. Inf. Technol. 2007. ICCGI 2007. Int. Multi-Conference*, pp. 31–31, 2007.

[19] S. Cantor, J. Kemp, and E. Maler, “Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0,” *OASIS Standard*, 2005. [Online]. Available: http://docs.oasis-open.org/security/saml/v2.0/. [Accessed: 15-Mar-2005].

[20] C. Kaler, M. Mcintosh, M. Goodner, and A. Nadalin, “Web Services Federation Language (WS-Federation) Version 1.2,” *OASIS Standard*, 2009. [Online]. Available: http://docs.oasis-open.org/wsfed/federation/v1.2/ws-federation.html. [Accessed: 22-May-2009].

[21] M. Behrendt, “IBM Cloud Computing Reference Architecture,” *Open Group submission*, 2011. [Online]. Available: http://www.opengroup.org/cloudcomputing/uploads/40/23840/CCRA.IBMSubmission.02282011.doc.

[22] C. Cadwalladr and E. Graham-Harrison, “Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach,” *The Guardian*, 2018. [Online]. Available: https://www.theguardian.com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election. [Accessed: 17-Mar-2018].

[23] M. Laurent and S. Bouzefrane, *Digital Identity Management*. London, UK: ISTE Press Ltd, 2015.

[24] BBC News, “Singapore personal data hack hits 1.5m, health authority says,” 2018. [Online]. Available: https://www.bbc.com/news/world-asia-44900507. [Accessed: 20-Jul-2018].

[25] F. Gonidis, I. Paraskakis, and D. Kourtesis, “Addressing the Challenge of Application Portability in Cloud Platforms,” in *Proceedings of the 7th South East European Doctoral Student Conference (DSC 2012)*, 2012, pp. 565–576.

[26] T. Sander and C. F. Tschudin, “Protecting Mobile Agents Against Malicious Hosts,” in *Mobile Agents and Security*, 1998, pp. 44–60.

[27] G. Gebel, “Authorization Performance Myth Busting,” 2010. [Online]. Available: http://analyzingidentity.com/2010/04/30/authorization-performance-myth-busting/. [Accessed: 30-Apr-2010].

[28] Cloud Security Alliance, “Security Guidance for Critical Areas of Focus in Cloud Computing V3.0,” *Cloud Secur. Alliance*, pp. 0–176, 2011.

[29] U. Habiba, R. Masood, M. Shibli, and M. Niazi, “Cloud identity management security issues & solutions: a taxonomy,” *Complex Adapt. Syst. Model.*, vol. 2, no. 1, pp. 1–37, 2014.

[30] Wikipedia, “Microsoft account --- Wikipedia{,} The Free Encyclopedia,” 2017. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Microsoft_account&oldid=771055380. [Accessed: 27-Apr-2017]

[31] “Liberty Alliance project.” [Online]. Available: http://www.projectliberty.org.

[32] R. Ranchal, B. Bhargava, L. Ben Othmane, L. Lilien, A. Kim, M. Kang, and M. Linderman, “Protection of Identity Information in Cloud Computing without Trusted Third Party,” in *2010 29th IEEE Symposium on Reliable Distributed Systems*, 2010, pp. 368–372.

[33] N. Grozev and R. Buyya, “Inter-Cloud architectures and application brokering: taxonomy and survey,” *Softw. Pract. Exp.*, vol. 44, no. 3, pp. 369–390, Mar. 2014.

[34] K. Rannenberg, J. Camenisch, and S. Ahmad, *Attribute-based Credentials for Trust*. Springer, 2015.

[35] J. Bellendorf and Z. Mann, “Cloud Topology and Orchestration Using TOSCA: A Systematic Literature Review: 7th IFIP WG 2.14 European Conference, ESOCC 2018, Como, Italy, September 12-14, 2018, Proceedings.” pp. 207–215, 2018.

[36] “Amazon DynamoDB Documentation,” *AmazonWS*. [Online]. Available: https://aws.amazon.com/documentation/dynamodb/. [Accessed: 20-May-2016].

[37] Z. Cai, L. Zhao, X. Wang, X. Yang, J. Qin, and K. Yin, “A Pattern-Based Code Transformation Approach for Cloud Application Migration,” in *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, 2015, pp. 33–40.

[38] D. Petcu, C. Cr\uaciun, M. Neagul, S. Panica, B. Di Martino, S. Venticinque, M. Rak, and R. Aversa, “Architecturing a Sky Computing Platform,” in *Proceedings of the 2010 International Conference on Towards a Service-based Internet*, 2011, pp. 1–13.

[39] D. Petcu, G. Macariu, S. Panica, and C. Crăciun, "Portable Cloud applications–From theory to practice," *Futur. Gener. Comput. Syst.*, vol. 29, no. 6, pp. 1417–1430, 2013.

[40] S. Frey, W. Hasselbring, and B. Schnoor, "Automatic Conformance Checking for Migrating Software Systems to Cloud Infrastructures and Platforms," *J. Softw. Evol. Process*, vol. 25, no. 10, pp. 1089–1115, 2013.

[41] T. Binz, U. Breitenbücher, O. Kopp, and F. Leymann, "Migration of enterprise applications to the cloud," *it - Inf. Technol.*, vol. 56, no. 3, pp. 106–111, 2014.

[42] A. Menychtas, C. Santzaridou, G. Kousiouris, T. Varvarigou, L. Orue-Echevarria, J. Alonso, J. Gorronogoitia, H. Bruneliere, O. Strauss, T. Senkova, B. Pellens, and P. Stuer, "ARTIST Methodology and Framework: A Novel Approach for the Migration of Legacy Software on the Cloud," in *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2013 15th International Symposium on*, 2013, pp. 424–431.

[43] J. Ejarque, A. Micsik, and R. M. Badia, "Towards Automatic Application Migration to Clouds," in *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, 2015, pp. 25–32.

[44] H. Brunelière, J. Cabot, G. Dupé, and F. Madiot,"MoDisco: a Model Driven Reverse Engineering Framework," *Inf. Softw. Technol.*, vol. 56, no. 8, pp. 1012–1032, Aug. 2014.

[45] N. Ferry, A. Rossini, F. Chauvel, B. Morin, and A. Solberg, "Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems," *Proc. IEEE Sixth Int. Conf. Cloud Comput.*, pp. 887–894, 2013.

[46] "JClouds." [Online]. Available: http://www.jclouds.org. [Accessed: 20-Jun-2005].

[47] S. Cimato, E. Damiani, F. Zavatarelli, and R. Menicocci, "Towards the Certification of Cloud Services," in *2013 IEEE Ninth World Congress on Services*, 2013, pp. 92–97.

[48] P. Membrey, K. C. C. Chan, C. Ngo, Y. Demchenko, and C. De Laat, "Trusted virtual infrastructure bootstrapping for on demand services," *Proc. - 2012 7th Int. Conf. Availability, Reliab. Secur. ARES 2012, Prague, 20-24 August 2012*, pp. 350–357.

[49] U. Lang, "OpenPMF SCaaS: Authorization as a service for cloud & SOA applications," *Proc. - 2nd IEEE Int. Conf. Cloud Comput. Technol. Sci. CloudCom 2010*, pp. 634–643, 2010.

[50] T. Waizenegger, M. Wieland, T. Binz, U. Breitenbücher, F. Haupt, O. Kopp, F. Leymann, B. Mitschang, A. Nowak, and S. Wagner, "Policy4TOSCA: A Policy-Aware Cloud Service Provisioning Approach to Enable Secure Cloud Computing," in *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*, 2013, pp. 360–376.

[51] H. Al-Aqrabi, L. Liu, J. Xu, R. Hill, N. Antonopoulos, and Y. Zhan, "Investigation of IT Security and Compliance Challenges in Security-as-a-Service for Cloud Computing," in *2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, 2012, pp. 124–129.

[52] Y. Sun, S. Nanda, and T. Jaeger, "Security-as-a-Service for Microservices-Based Cloud Applications," in *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, 2015, pp. 50–57.

[53] "OpenStack." [Online]. Available: https://www.openstack.org/. [Accessed: 19-Mar-2018].

[54] N. Macdonald, "Securing the Next-Generation Virtualized Data Center," 2010.

[55] M. Almorsy, J. Grundy, and A. S. Ibrahim, "TOSSMA: A Tenant-Oriented SaaS Security Management Architecture," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, 2012, pp. 981–988.

[56] R. Johansen, S. Spangenberg, and P. Sestoft, "Yiihaw . NET aspect weaver usage guide," *IT University of Copenhagen, Denmark*, 2007. .

[57] E. W. Dijkstra, *Notes on structured programming*. 1972.

[58] Y. Demchenko, C. Ngo, C. De Laat, D. R. Lopez, A. Morales, C. de Laat, D. R. Lopez, A. Morales, and J. García-Espín, "Security Infrastructure for Dynamically Provisioned Cloud Infrastructure Services," in *Privacy and Security for Cloud Computing SE - 5*, S. Pearson and G. Yee, Eds. Springer London, 2013, pp. 167–210.

[59] Wikipedia contributors, "Enterprise service bus --- {Wikipedia}{,} The Free Encyclopedia," 2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Enterprise_service_bus&oldid=924139487. [Accessed: 19-Mar-2016].

[60] K. Lawrence, C. Kaler, A. Nadalin, M. Goodner, and M. Gudgin, "WS-Trust 1.4," *Oasis Standard*, 2009. [Online]. Available: http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/os/ws-trust-1.4-spec-os.html.

[61] E. Birrell and F. B. Schneider, "Federated identity management systems: A privacy-based characterization," *IEEE Secur. Priv.*, vol. 11, no. 5, pp. 36–48, 2013.

[62] F. Hirsch, R. Philpott, and E. Maler, "Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0," *OASIS Standard*, 2005. [Online]. Available: http://docs.oasis-open.org/security/saml/v2.0/.

[63] J. De Groot, "Safeguarding Privacy in Trust Negotiation."

[64] M. Lizar and D. Turner, "Consent Receipt Specification v.1.1.0," *Kantara Initiative Technical Specification*, 2018. [Online]. Available: https://kantarainitiative.org/file-downloads/consent-receipt-specification-v1-1-0/. [Accessed: 20-Feb-2018].

[65] C. Adams and S. Farrell, “Internet X.509 Public Key Infrastructure Certificate Management Protocols and Certificate Revocation List (CRL) Profile,” *Internet Engineering Task Force (IETF)*, 1999. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2510.txt.

[66] G. Neven, “A Quick Introduction to Anonymous Credentials,” *IBM Zürich Research Laboratory*, 2008. [Online]. Available: https://idemix.wordpress .com/2009/08/18/quick-intro-to-credentials/.

[67] I. Damgard, “Commitment Schemes and Zero-Knowledge Protocols,” *Proc. Lect. Data Secur.*, pp. 63–86, 1999.

[68] D. Chaum, “Security without identification: transaction systems to make big brother obsolete,” *Commun. ACM*, vol. 28, no. 10, pp. 1030–1044, 1985.

[69] J. Camenisch and A. Lysyanskaya, “An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation.,” *Adv. Cryptol. - EUROCRYPT 2001, Int. Conf. Theory Appl. Cryptogr. Tech. Innsbruck, Austria, May 6-10, 2001*, vol. 2045, pp. 93–118, 2001.

[70] C. Paquin, “U-Prove Technology Overview V1.1 (Revision 2).” Microsoft, Apr-2013.

[71] B. Campbell, C. Mortimore, and M. Jones, “Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants,” *Internet Engineering Task Force (IETF)*, 2015. [Online]. Available: https://tools.ietf.org/html/rfc7522. [Accessed: 31-May-2015].

[72] D. Hardt, “The OAuth 2.0 Authorization Framework,” *Internet Engineering Task Force (IETF)*, 2012. [Online]. Available: http://tools.ietf.org/ html/rfc6749.

[73] E. B. A. Bhargav-Spantzel, A.C. Squicciarini, R. Xue, “Practical Identity Theft Prevention using Aggregated Proof of Knowledge,” West Lafayette, 2001.

[74] E. Bertino, W. Lafayette, F. Paci, and R. Ferrini, “Privacy-preserving Digital Identity Management for Cloud Computing,” *Identity*, vol. 32, pp. 1–7, 2009.

[75] N. Guo, T. Gao, B. Zhang, R. Fernando, and E. Bertino, “Aggregated Privacy-Preserving Identity Verification for Composite Web Services,” in *2011 IEEE International Conference on Web Services*, 2011, pp. 692–693.

[76] D. Chaum and E. Van Heyst, “Group Signatures,” in *Proceedings of the 10th Annual International Conference on Theory and Application of Cryptographic Techniques*, 1991, pp. 257–265.

[77] S. S. M. Chow, Y.-J. He, L. C. K. Hui, and S. M. Yiu, “SPICE -- Simple Privacy-Preserving Identity-Management for Cloud Environment,” in *Applied Cryptography and Network Security: 10th International Conference, ACNS 2012, Singapore, June 26-29, 2012. Proceedings*, F. Bao, P. Samarati, and J. Zhou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 526–543.

[78] L. Malina, J. Hajny, P. Dzurenda, and V. Zeman, “Privacy-preserving security solution for cloud services,” *J. Appl. Res. Technol.*, vol. 13, no. 1, pp. 20–31, 2015.

[79] S. Cantor and I. Kemp, “Assertions and protocols for the oasis security assertion markup language,” *OASIS Stand. (March …*, no. March, pp. 1–86, 2005.

[80] D. W. Chadwick and K. Fatema, “A Privacy Preserving Authorisation System for the Cloud,” *J. Comput. Syst. Sci.*, vol. 78, no. 5, pp. 1359–1373, Sep. 2012.

[81] M. C. Mont, S. Pearson, and P. Bramhall, “Towards accountable management of identity and privacy: sticky policies and enforceable tracing services,” *14th Int. Work. Database Expert Syst. Appl. 2003. Proceedings.*, 2003.

[82] D. Boneh and M. Franklin, “Identity-Based Encryption from the Weil Pairing,” *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, 2003.

[83] M. Beiter, M. C. Mont, L. Chen, and S. Pearson, “End-to-end policy based encryption techniques for multi-party data management,” *Comput. Stand. Interfaces*, vol. 36, no. 4, pp. 689–703, 2014.

[84] L. Ben Othmane, “Active Bundles for Protecting Confidentiality of Sensitive Data Throughout Their Lifecycle,” Western Michigan University, Kalamazoo, MI, USA, 2010.

[85] K. Bendiab, N. Kolokotronis, S. Shiaeles, and S. Boucherkha, “WiP: {A} Novel Blockchain-Based Trust Model for Cloud Identity Management,” in *2018 {IEEE} 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress, DASC/PiCom/DataCom/CyberSciTe*, 2018, pp. 724–729.

[86] T. Mikula and R. H. Jacobsen, “Identity and Access Management with Blockchain in Electronic Healthcare Records,” *2018 21st Euromicro Conf. Digit. Syst. Des.*, pp. 699–706, 2018.

[87] J.-W. Byun and N. Li, “Purpose Based Access Control for Privacy Protection in Relational Database Systems,” *VLDB J.*, vol. 17, no. 4, pp. 603–619, Jul. 2008.

[88] J. Kiernan, R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, “Hippocratic Databases,” in *Proceedings of the 28th International Conference on Very Large Data Bases*, 2002, vol. 4, no. 1890, pp. 143–154.

[89] J.-W. Byun, E. Bertino, and N. Li, “Purpose based access control of complex data for privacy protection,” *SACMAT ’05 Proc. tenth ACM Symp. Access Control Model. Technol.*, pp. 102–110, 2005.

[90] N. Yang and H. Barringer, “A purpose-based access control model,” *Inf. Assur.*, vol. 1, pp. 51–58, 2007.

[91] Q. Ni, E. Bertino, J. Lobo, C. Brodie, C.-M. Karat, J. Karat, and A. Trombeta, "Privacy-aware role-based access control," *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 3, pp. 1–31, 2010.

[92] M. E. Kabir, H. Wang, and E. Bertino, "A role-involved purpose-based access control model," *Inf. Syst. Front.*, vol. 14, no. 3, pp. 809–822, Jul. 2012.

[93] A. C.-C. Yao, "How to Generate and Exchange Secrets," in *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, 1986, pp. 162–167.

[94] J. T. Mcdonald, "Enhanced Security for Mobile Agent Systems," Florida State University, Tallahassee, FL, USA, 2006.

[95] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," in *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, 2009, pp. 169–178.

[96] N. P. Smart and F. Vercauteren, "Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes," in *Proceedings of the 13th International Conference on Practice and Theory in Public Key Cryptography*, 2010, pp. 420–443.

[97] P. Snyder, "Yao's Garbled Circuits: Recent Directions and Implementations," 2012.

[98] B. Kreuter, A. Shelat, and C.-H. Shen, "Billion-gate Secure Computation with Malicious Adversaries," in *Proceedings of the 21st USENIX Conference on Security Symposium*, 2012, p. 14.

[99] A.-R. Sadeghi, T. Schneider, and M. Winandy, "Token-Based Cloud Computing," in *Trust and Trustworthy Computing: Third International Conference, TRUST 2010, Berlin, Germany, June 21-23, 2010. Proceedings*, A. Acquisti, S. W. Smith, and A.-R. Sadeghi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 417–429.

[100] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based Encryption for Fine-grained Access Control of Encrypted Data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, 2006, pp. 89–98.

[101] D. Boneh, A. Sahai, and B. Waters, "Functional Encryption: Definitions and Challenges," in *Theory of Cryptography: 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*, Y. Ishai, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 253–273.

[102] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," in *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, 2007, pp. 321–334.

[103] J. Katz, A. Sahai, and B. Waters, "Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products," in *Proceedings*

of the Theory and Applications of Cryptographic Techniques 27th Annual International Conference on Advances in Cryptology, 2008, pp. 146–162.

[104] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption," in *Advances in Cryptology -- EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 -- June 3, 2010. Proceedings*, H. Gilbert, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 62–91.

[105] T. Okamoto and K. Takashima, "Hierarchical Predicate Encryption for Inner-Products," in *Advances in Cryptology -- ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, M. Matsui, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 214–231.

[106] T. Okamoto and K. Takashima, "Adaptively Attribute-Hiding (Hierarchical) Inner Product Encryption," *IACR Cryptol. ePrint Arch.*, vol. 2011, p. 543, 2011.

[107] Y. Kawai and K. Takashima, "Predicate- and Attribute-Hiding Inner Product Encryption in a Public Key Setting," in *Pairing-Based Cryptography -- Pairing 2013: 6th International Conference, Beijing, China, November 22-24, 2013, Revised Selected Papers*, Z. Cao and F. Zhang, Eds. Cham: Springer International Publishing, 2014, pp. 113–130.

[108] M. Li, S. Yu, N. Cao, and W. Lou, "Authorized Private Keyword Search over Encrypted Data in Cloud Computing," in *Proceedings of the 2011 31st International Conference on Distributed Computing Systems*, 2011, pp. 383–392.

[109] A. Sahai and H. Seyalioglu, "Dynamic credentials and ciphertext delegation for attribute-based encryption," in *in Proceedings of the 32nd Annual International Cryptology Conference: Advances in Cryptology - CRYPTO 2012*, 2012, pp. 199–217.

[110] K. Yang and X. Jia, "Expressive, Efficient, and Revocable Data Access Control for Multi-Authority Cloud Storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1735–1744, Jul. 2014.

[111] S. Jahid, P. Mittal, and N. Borisov, "EASiER: Encryption-based Access Control in Social Networks with Efficient Revocation," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, 2011, pp. 411–415.

[112] A. Lewko and B. Waters, "Unbounded HIBE and Attribute-Based Encryption," in Advances in Cryptology -- EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings, K. G. Paterson, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 547–567.

[113] Y. Rouselakis and B. Waters, “Practical Constructions and New Proof Methods for Large Universe Attribute-based Encryption,” in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, 2013, pp. 463–474.

[114] A. Lewko and B. Waters, “Decentralizing Attribute-Based Encryption,” in *Advances in Cryptology -- EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, K. G. Paterson, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 568–588.

[115] Y. Rouselakis and B. Waters, “Efficient Statically-Secure Large-Universe Multi-Authority Attribute-Based Encryption,” in *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, R. Böhme and T. Okamoto, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 315–332.

[116] K. Brown, *The .NET developer’s guide to Windows security*. Addison-Wesley, 2004.

[117] E. Spear, “Understanding Identity Propagation,” in *Using Oracle Java Cloud Service - SaaS Extension, Release 16.3*, Oracle® Cloud, 2016.

[118] S. Cantor, “SAML V2.0 Condition for Delegation Restriction,” *OASIS Committee Specification 01*, 2009. [Online]. Available: http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-delegation-cs-01.html. [Accessed: 31-Aug-2016].

[119] P. Siriwardena, “Advanced API Security,” 2014.

[120] P. Hunt, K. Grizzle, M. Ansari, E. Wahlstroem, and C. Mortimore, “System for Cross-domain Identity Management: Protocol,” *Internet Engineering Task Force (IETF)*, 2015. [Online]. Available: http://www.ietf.org/rfc/rfc7644.txt.

[121] S. Gnaniah, “WSO2 Identity Server Documentation,” *WSO2 Documentation*, 2015. [Online]. Available: http://docs.wso2.com/. [Accessed: 19-Mar-2018].

[122] F. Chong, “Identity and Access,” in *SOA in the Real World*, Microsoft Developer Network.

[123] Oracle, “The Java EE 6 Tutorial,” Sun Microsystems, Inc., Santa Clara, 2013.

[124] M. Kalali, *GlassFish Security*. Birmingham, UK: Packt Publishing Ltd., 2010.

[125] Oracle, “Oracle GlassFish Server 3.1 Application Deployment Guide,” 2011. [Online]. Available: https://docs.oracle.com/cd/E18930_01/pdf/821-2417.pdf.

[126] J. Lowy, *Programming WCF Services*. O’Reilly Media, Inc., 2010.

[127] J. Hogg, D. Smith, F. Chong, D. Taylor, L. Wall, P. Slater, T. Hollander, and W. Kozaczynski, *Web Service Security (Scenarios, Patterns, and Implementation Guidance for Web Services Enhancements (WSE) 3.0)*. Microsoft Corporation, 2005.

[128] C. Steel, *Core Security Patterns: Best Practices and Strategies for J2EE, Web Services, and Identity Management*. New Jersey, United States: Prentice Hall, 2005.

[129] Oracle, "Metro User Guide," *GlassFish documentation*. [Online]. Available: https://metro.java.net/guide/. [Accessed: 06-Sep-2016].

[130] Wikipedia contributors, "Kerberos (protocol)," *Wikipedia, The Free Encyclopedia*, 2016. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Kerberos_(protocol)&oldid=736925661. [Accessed: 25-Sep-2016].

[131] A. Nadalin, Chris Kaler, Ronald Monzillo, and Phillip Hallam-Baker, "Web Services Security X.509 Certificate Token Profile 1.1," *OASIS Standard Specification*, 2006. [Online]. Available: http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-x509TokenProfile.pdf.

[132] M. J. Fowler, *Patterns of Enterprise Application Architecture*. Addison Wesley, 2002.

[133] S. Chapin, D. Faatz, S. Jajodia, and A. Fayad, "Consistent policy enforcement in distributed systems using mobile policies," *Data Knowl. Eng.*, vol. 43, no. 3, pp. 261–280, 2002.

[134] M. Craig, G. Hirayama, and C. Lee, "OpenAM Java EE Policy Agent User's Guide version 3.5," *Forgerock documentation*, 2016. [Online]. Available: https://backstage.forgerock.com. [Accessed: 17-Oct-2016].

[135] Cafesoft, "Cams Servlet Filter Web Agent Guide," *Cams documentation*. [Online]. Available: http://www.cafesoft.com/products/cams/wa/servletFilter/docs32/index.html. [Accessed: 18-Oct-2016].

[136] Onelogin, "Developing with Web Access Management (WAM)," *Onelogin developers documentation*. [Online]. Available: https://developers.onelogin.com/wam. [Accessed: 17-Oct-2016].

[137] Oracle, "Java Authentication and Authorization Service (JAAS): LoginModule Developer's Guide," *Java Platform Standard Edition 8 Documentation*. [Online]. Available: https://docs.oracle.com/javase/8/docs/technotes/guides/security/jaas/JAASLMDevGuide.html. [Accessed: 18-Oct-2016].

[138] Oracle, "Java Authentication and Authorization Service (JAAS) Reference Guide," *Java Platform Standard Edition 8 Documentation*. [Online]. Available: https://docs.oracle.com/javase/8/docs/technotes/guides/security/jaas/JAASRefGuide.html. [Accessed: 20-Oct-2016].

[139] B. Pontarelli, “J2EE security: Container versus custom,” *JavaWorld*. [Online]. Available: http://www.javaworld.com/article/2072850/java-security/j2ee-security--container-versus-custom.html. [Accessed: 05-Oct-2016].

[140] A. Patel, M. McRoberts, and M. Crenshaw, “Identity propagation in N-tier systems,” in *MILCOM 2009 - 2009 IEEE Military Communications Conference*, 2009, pp. 1–5.

[141] S. Guilhen, “Issuing And Propagating SAML Assertions Within JBoss AS,” *JBoss Developer Documentation*, 2010. [Online]. Available: https://developer.jboss.org/wiki/IssuingAndPropagatingSAMLAssertionsWithinJBossAS.

[142] Microsoft, “Windows Identity Foundation,” *.NET Development*. [Online]. Available: https://msdn.microsoft.com/en-us/library/ee748484.aspx. [Accessed: 26-Oct-2016].

[143] V. Bertocci, *Programming Windows Identity Foundation*. Redmond, Washington: Microsoft Press, 2011.

[144] N. Klingenstein, “Attribute Storage,” *Shibboleth Concepts*, 2008. [Online]. Available: https://wiki.shibboleth.net/confluence/display/CONCEPT/Attribute Storage. [Accessed: 20-Mar-2008].

[145] WSO2, “The WS02 SOA Security Gateway Solution,” *WSO2 White paper*. [Online]. Available: http://wso2.com/whitepapers/. [Accessed: 26-Aug-2016].

[146] P. Bryan, M. Craig, J. Nelson, G. Sauthier, and J. Henry, “OpenIG Gateway Guide version 5.0.0,” *Forgerock documentation*, 2016. [Online]. Available: https://forgerock.org/openig/doc/bootstrap/gateway-guide/index.html.

[147] D. Palma, M. Rutkowski, and T. Spatzier, “TOSCA Simple Profile in YAML Version 1.0,” *OASIS Standard*, 2016. [Online]. Available: http://docs.oasis-open.org/tosca/TOSCASimple-.

[148] K. Lawrence and C. Kaler, “WS-SecurityPolicy 1.2,” *Oasis*, 2007. [Online]. Available: http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.pdf. [Accessed: 25-Apr-2012].

[149] A. De Caro and V. Iovino, “jPBC: Java pairing based cryptography,” in *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*, 2011, pp. 850–855.

[150] B. Lynn, *On the Implementation of Pairing-based Cryptosystems*. Stanford University, 2007.

[151] N. Attrapadung, G. Hanaoka, K. Ogawa, G. Ohtake, H. Watanabe, and S. Yamada, “Attribute-Based Encryption for Range Attributes,” in *Security and Cryptography for Networks: 10th International Conference, SCN 2016, Amalfi, Italy, August 31 -- September 2, 2016, Proceedings*, V. Zikas and R. De Prisco, Eds. Cham: Springer International Publishing, 2016, pp. 42–61.

[152] J. Daemen and V. Rijmen, “AES Proposal: Rijndael,” 1999. [Online]. Available: http://www.cryptosoft.de/docs/Rijndael.pdf. [Accessed: 19-Mar-2017].

[153] M. Morales-Sandoval and A. Diaz-Perez, “DET-ABE: A Java API for Data Confidentiality and Fine-Grained Access Control from Attribute Based Encryption,” in *Information Security Theory and Practice: 9th IFIP WG 11.2 International Conference, WISTP 2015, Heraklion, Crete, Greece, August 24-25, 2015. Proceedings*, R. N. Akram and S. Jajodia, Eds. Cham: Springer International Publishing, 2015, pp. 104–119.

[154] D. Boneh, “Pairing-Based Cryptography: Past, Present, and Future,” in *Advances in Cryptology -- ASIACRYPT 2012*, 2012, p. 1.

[155] S. D. Galbraith, K. G. Paterson, and N. P. Smart, “Pairings for Cryptographers,” *Discret. Appl. Math.*, vol. 156, no. 16, pp. 3113–3121, Sep. 2008.

[156] E. Barker, “Recommendation for Key Management – Part 1: General,” *NIST Special Publication 800-57*, 2016. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf. [Accessed: 01-Jan-2017].

[157] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Commun. ACM*, 1978.

[158] M. Morales-Sandoval, J. L. Gonzalez-Compean, A. Diaz-Perez, and V. J. Sosa-Sosa, “A pairing-based cryptographic approach for data security in the cloud,” *Int. J. Inf. Secur.*, vol. 17, no. 4, pp. 441–461, Aug. 2018.

[159] “AmazonWS.” [Online]. Available: https://aws.amazon.com/. [Accessed: 19-Mar-2018].

[160] “Alien4Cloud version 1.4.” [Online]. Available: https://alien4cloud.github.io/. [Accessed: 19-Mar-2018].

[161] “Cloudify Documenation.” [Online]. Available: https://docs.cloudify.co/. [Accessed: 20-May-2016].

[162] “Cloud-Init Documentation.” [Online]. Available: http://cloudinit.readthedocs.io. [Accessed: 17-May-2018].

[163] “Apache Synapse Enterprise Service Bus (ESB).” [Online]. Available: http://synapse.apache.org/. [Accessed: 19-Apr-2018].

[164] F. Leymann, M. Rutkowski, A. Hohl, M. Waschke, and P. Zhang, “Topology and Orchestration Specification for Cloud Applications (TOSCA) Primer Version 1.0,” *OASIS Committee Note Draft 01*, 2013. [Online]. Available: http://docs.oasis-open.org/tosca/tosca-primer/v1.0/cnd01/tosca-primer-v1.0-cnd01.html. [Accessed: 31-Jan-2013].

[165] R. Monzillo, “Java Authentication SPI for Containers (JSR-196) Version 1.1,” *Java Community Process*, 2013. [Online]. Available: https://www.jcp.org/en/jsr/detail?id=196.

[166] P. S. L. M. Barreto and M. Naehrig, “Pairing-Friendly Elliptic Curves of Prime Order,” in *Selected Areas in Cryptography*, 2006, pp. 319–331.

[167] Wikipedia contributors, “Edge computing --- {Wikipedia}{,} The Free Encyclopedia,” 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Edge_computing&oldid=834289034.

[168] G. Coker, J. Guttman, P. Loscocco, A. Herzog, J. Millen, B. O’Hanlon, J. Ramsdell, A. Segall, J. Sheehy, and B. Sniffen, “Principles of remote attestation,” *Int. J. Inf. Secur.*, vol. 10, no. 2, pp. 63–81, Jun. 2011.

[169] S. Newman, *Building Microservices*. Oreilly, 2015.

Appendix A

A1. Linear Secret Sharing Schemes

Figure 59 shows an example that converts a disclosure policy from a Boolean formula to an LSSS Matrix. The Boolean formula consists of two operators (i.e., AND, OR) and a set of key attributes (e.g., A, B, C, D). The first step transforms the Boolean formula into a Boolean tree. In the tree presentation, one parent node has exact two children nodes, which are either a Boolean operator or a key attribute. The leaf nodes of the tree are the set of all key attributes. The second step transforms the Boolean tree into an LSSS Matrix by following the algorithm in [107].

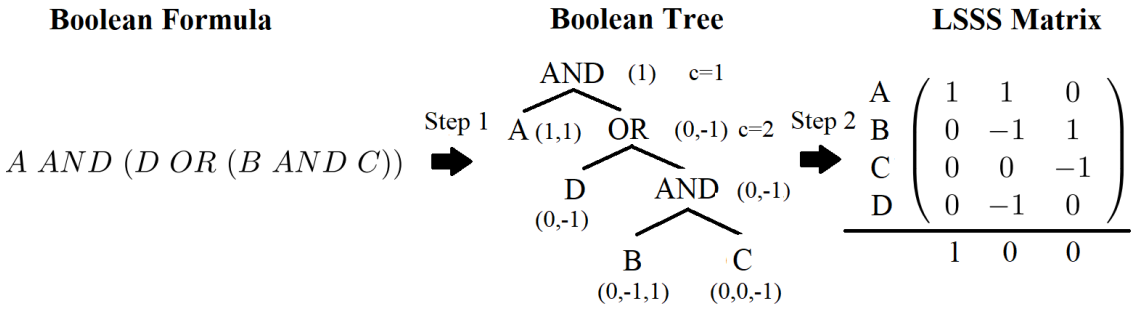

Figure 59: LSSS Matrix generation

The algorithm in step 2 is summarised as follows: The algorithm starts with a global counter variable c, which is initialized to “1” and labels the root node of the tree with the vector (1) (e.g., the root node in Figure 59 has the vector 1). Then the algorithm goes down the levels of the tree and labels each node with a vector determined by the vector assigned to its parent node. If the parent node is an OR gate labelled by the vector v, its children nodes inherit the same vector v. In this example, the key attribute D get the vector (0,-1) from its parent node (i.e., an OR gate).

If the parent node is an AND gate, the algorithm labels the left child node as (v, 1) and label the right child node as (0, …, 0_c, -1), where (0, …, 0_c) denotes the zero vector of length c. Then the algorithm increments the value of c by 1. In this example, the parent vector of B and C is an AND gate with vector (0, -1). Therefore, B and C have the vector (0, -1, 1) and (0, 0, -1), respectively.

Once the algorithm has finished labelling the entire tree, the vectors of the leaf nodes form the rows of the LSSS matrix. If a set of key attributes (e.g., A and D) satisfy the Boolean formula, the sum of their vector results in the vector 1.

A2. Encode a numerical range

Figure 60 shows an example to encode a numerical range to a Boolean formula. The encoding transforms a numerical range (e.g., from 1 to 8) to a segment tree as follows:

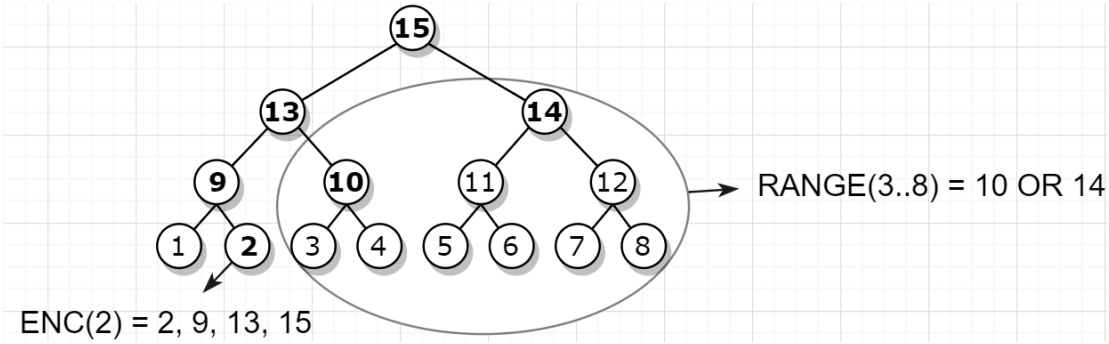

Figure 60: Segment tree encoding

In the segment tree above, the leaf nodes of the tree are a set of all key attributes (e.g., 1, 2, 3, 4, 5, 6, 7, and 8). Every two children nodes have exactly one parent node that has a higher number. A numerical range by is encoded finding the least common nodes of the leaf nodes. In the given example, the two ranges (from 3 to 4) and (from 5 to 8) have two least common nodes 10 and 14, respectively. Therefore, the encoding presents this range by the Boolean formula (10 OR 14).

The process encodes a key attribute by drawing the path from the root to the leaf node and collecting all these nodes on the way. In this example, the process encodes the key attribute 2 by a set of four nodes (2, 9, 13, and 15). This set does not have any common nodes with the range presentation (10, 14). Therefore, the key attribute 2 does not belong to the given range.

A3. Bilinear maps

Let G and $\mathrm{G_T}$ be two multiplicative cyclic groups of prime order p. Let g be a generator of G and *e* be a bilinear map, *e:* $\mathrm{G} \times \mathrm{G} \rightarrow \mathrm{G_T}$. The bilinear map *e* has the following properties:

4. Bilinearity: for all u, v ∈ G and a, b ∈ Z_p, $e(u^a, v^b) = e(u, v)^{ab}$.
5. Non-degeneracy: $e(g, g) \neq 1$.

G is a bilinear group if the group operation in G and the bilinear map e: $\mathrm{G} \times \mathrm{G} \rightarrow \mathrm{G_T}$ are both efficiently computable. Notice that the map e is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

A4. Multi-authority ABE scheme implementation

Given:

- A bilinear group G of prime order p with generator g.
- A function H that maps a global identity GID (string) to an element of G.
- A function F that maps a key attribute (string) to an element of G.

Setup global parameter: $\mathrm{GP} = \{\mathrm{p}, \mathrm{G}, \mathrm{g}, \mathrm{H}, \mathrm{F}\}$

Setup an authority θ:

- Choose two random exponents $\alpha_\theta, y_\theta \leftarrow \mathrm{Z_p}$
- Master public key $\mathrm{PK}_\theta = \{\mathrm{e}(\mathrm{g},\mathrm{g})^{\alpha_\theta},\ \mathrm{g}^{y_\theta}\}$
- Master secret keys $\mathrm{SK}_\theta = \{\alpha_\theta, y_\theta\}$

Generate a secret key for an attribute u:

- Choose a random $t \leftarrow Z_p$
- $K_{GID,u} = g^{\alpha_\theta} H(GID)^{y_\theta} F(u)^t$
- $K'_{GID,u} = g^t$

Encrypt a message M with an LSSS Matrix A (l rows, n columns):

- Choose random elements $z, v_2 \dots, v_n, w_2, \dots, w_n \leftarrow Z_p$

  Create a secret vector $\vec{v} = (z, v_2 \dots, v_n)$ and a zero vector $\vec{w} = (0, w_2, \dots, w_n)$
- z is denoted as the decryption key. Then for each row x of the matrix, the encryption calculates the secret share λ_x of z as well as the share w_x of zero:

  $\lambda_x = \langle A_x, \vec{v} \rangle$, and $w_x = \langle A_x, \vec{w} \rangle$
- For each row x of matrix A, choose a random $t^x \leftarrow Z_p$ and compute the ciphertext as:
  - $CT = M.e(g,g)^z$
  - $C_{1,x} = e(g,g)^{\lambda_x} e(g,g)^{\alpha_{\theta,x} t_x}, C_{2,x} = g^{-t_x}, C_{3,x} = g^{y_{\theta,x} t_x} g^{w_x}, C_{4,x} = F(x)^{t_x}$

Decrypt ciphertext CT:

- For a set of attributes that satisfy the disclosure policy:

  Compute $e(g,g)^z = \prod_x (C_{1,x}.e(K_{GID,x}, C_{2,x}).e(H(GID), C_{3,x}).e(K'_{GID,x}, C_{4,x}))^{c_x}$
- Compute plaintext $M = CT/e(g,g)^z$

Appendix B

List of own publications

The research has published the following publications. Work in [1] described the missing gap of IDaaS in addition to a traditional IDM and proposed a draft architecture for IDaaS. It reflects Chapter 2 and the first part of Chapter 3 (Section 3.1). Work in [2] reviewed the adaptation of AAI for the cloud environment and proposed a draft concept of the security topology. It reflects Section 3.2, Chapter 5, and Chapter 6. On the other hand, work in [3] reviewed the research challenge of privacy-preserving user identity and proposed a draft concept of Purpose-based Encryption. It reflects Chapter 4 and Section 7.2. Work in [4] refined the concept of Purposed-based Encryption and evaluated the experiment results. It reflects Section 7.2 and Section 8.1. Work in [5] refined the concept of security topology and evaluated the experiment results. It reflects Section 7.3 and Section 8.2. Finally, work in [6] summarised all 5 publications in a journal article.

[1] T. H. Vo, W. F. Fuhrmann, and K. P. Fischer-Hellmann, “Identity-as-a-Service (IDaaS): {A} Missing Gap for Moving Enterprise Applications in Inter-Cloud,” in Eleventh International Network Conference, {INC} 2016, Frankfurt, Germany, July 19-21, 2016. Proceedings, 2016, pp. 121–126. ISBN: 978-1-84102-428-8.

[2] T. H. Vo, W. Fuhrmann, and K. P. Fischer-Hellmann, “How to adapt authentication and authorization infrastructure of applications for the cloud,” Proc. - 2017 IEEE 5th Int. Conf. Future Internet Things Cloud, FiCloud 2017, vol. 2017-Januar, pp. 54–61, 2017. DOI:10.1109/FiCloud.2017.14.

[3] T. H. Vo, W. Fuhrmann, and K. P. Fischer-Hellmann, “Privacy-preserving user identity in Identity-as-a-Service,” in 21st Conference on Innovation in Clouds, Internet and Networks, ICIN 2018, 2018, pp. 1–8. DOI: 10.1109/ICIN.2018.8401613.

[4] T. H. Vo, W. Fuhrmann, K. Fischer-Hellmann, and S. Furnell, “Efficient Privacy-preserving User Identity with Purpose-based Encryption,” in 2019

International Symposium on Networks, Computers and Communications (ISNCC), 2019, pp. 1–8. DOI: 10.1109/ISNCC.2019.8909174.

[5] T. H. Vo, W. Fuhrmann, K. P. Fischer-Hellmann, and S. Furnell, “Automated trust negotiation for cloud applications in identity-as-a-service,” in Proceedings - 2019 International Conference on Advanced Communication Technologies and Networking, CommNet 2019, 2019, pp. 1–8. DOI: 10.1109/COMMNET.2019.8742384.

[6] T. H. Vo, W. Fuhrmann, K.-P. Fischer-Hellmann, and S. Furnell, “Identity-as-a-Service: An Adaptive Security Infrastructure and Privacy-Preserving User Identity for the Cloud Environment,” Future Internet, vol. 11, no. 5, 2019. DOI: 10.3390/fi11050116.