Secure idenTity crOss boRder linKed - eLearning and Academic Qualifications Pilot at IST

Simon Filipe Pacheco Esposito

Thesis to obtain the Master of Science Degree in

Telecommunications and Informatics Engineering

Supervisors: Prof. Carlos Nuno da Cruz Ribeiro
Prof. Fernando Henrique Côrte-Real Mira da Silva

Examination Committee

Chairperson: Prof. Paulo Jorge Pires Ferreira
Supervisor: Prof. Carlos Nuno da Cruz Ribeiro
Member of the Committee: Prof. André Ferreira Ferrão Couto e Vasconcelos

June 2015

Acknowledgments

I begin by expressing my sincere gratitude to Professor Carlos Ribeiro for giving me the opportunity to work with him in this project and for his guidance along the way. It was a great experience.
I also thank Diogo Antunes for his contributions in the development of some of this work and all the DSI staff for their availability.
I am deeply thankful for my parent's support and encouragement that lead me to where I am now.
And finally, I would like to say thank you to everyone that accompanied me in this journey, for all the friendship and support you have provided throughout these years: Jorge Santos, Jorge Duarte, Pedro Gavinho, Rui Pires, Alexandre Campos, Gonçalo Pestana, Carl Kristensen, David Nogueira, Joana Marques and everyone else that positively influenced me during this time at IST.

Abstract

The demand for federated identity management systems is growing. With the current thrive of online services on the web, the interest of governments in offering online public services for citizens and organizations is on the rise. This creates the need for digital identity management systems that can provide a high level of authentication assurance while keeping the risk of using such services at a minimum for all the involved stakeholders. The Secure idenTity acrOss boRders linKed (STORK) Large Scale Pilot (LSP) has been presented as a solution to this demand, by creating a unified European electronic identification and authentication area between the Member States (MSs) of the European Union (EU). Within the scope of the academic STORK LSP, in this work we will present the proposed solution and implementation of an eLearning platform that uses the STORK infrastructure for authentication and cross-federation attribute exchange; and Trusted Attribute Display Service (TADS), a service for the dematerialization and materialization of digitally signed documents between European federations.

Keywords

STORK, Identity Management System, Federated Identity Management, Federated Attribute Exchange, eGovernment services

Resumo

A necessidade de sistemas de gestão de identidades federadas está em ascensão. Com a atual prosperidade de serviços online disponíveis na Internet, cada vez mais, governos e organizações têm interesse em oferecer os seus próprios serviços, não só devido à diminuição de custos associada à automatização de processos de negócio e burocráticos, mas também à inerente melhoria da experiência para os utilizadores que usam esses serviços.
Isto cria procura por sistemas de gestão de identidades digitais federadas que possam garantir um alto nível de segurança, quer para os utilizadores quer para os provedores de serviços e que forneça elevada confiança na autenticação para que o risco associado à utilização dos serviços oferecidos seja mínimo para todas as entidades envolvidas na federação. O projeto STORK foi introduzido como uma solução para essa procura através da criação de um espaço europeu unificado de identificação e autenticação eletrónica entre os estados membros da União Europeia.
No âmbito do piloto académico do STORK, neste trabalho vamos apresentar a solução e implementação proposta para uma plataforma de eLearning que faz uso da infraestrutura do projeto para autenticação federada entre estados membros e troca de atributos credenciados entre estes; e ainda o TADS, um serviço para a desmaterialização e materialização de documentos com informação credenciada, verificáveis através de assinaturas digitais.

Palavras Chave

STORK, Sistemas de Gestão de Identidades, Identidades Federadas, Identidades Electrónicas

Contents

List of Figures

Acronyms

AMA Agência para a Modernização Administrativa

API Application Programming Interface

AP Attribute Provider

CAS Central Authentication Service

CA Certificate Authority

ECC European Citizen Card

EDS European Diploma Supplement

eID Electronic Identity

EU European Union

HSM Hardware Security Module

IdM Identity Management System

IdP Identity Provider

IST Instituto Superior Técnico

JSON JavaScript Object Notation

LDAP Lightweight Directory Access Protocol

LoA Level of Assurance

LSP Large Scale Pilot

MCM Microsoft CardSpace Metasystem

MS Member State

mTAN Mobile Transaction Authentication Number

MVC Model-View-Controller

NIC Numero de Identificacao do Cidadao

PDF Portable Document Format

PEPS Pan-European Proxy Service

PIAP Plataforma de Interoperabilidade da Administracao Publica

PId Persistent Identifier

PIN Personal Identification Number

PKI Public Key Infrastructure

PKI Public Key Infrastructure

REST Representational State Transfer

SAML Security Assertion Markup Language

SCAP Sistema de Certificação de Atributos Profissionais

SC SmartCard

SIdP Shibboleth Identity Provider

SIM Subscriber Identity Module

SMS Short Message Service

SOAP Simple Object Access Protocol

SP Service Provider

SSO Single Sign On

SSP Shibboleth Service Provider

STORK Secure idenTity acrOss boRders linKed

TAAS Trusted Attribute Aggregation Service

TADS Trusted Attribute Display Service

TOR The Onion Routing

UI User Interface

URI Unique Resource Identifier

VID Virtual Identity

VPN Virtual Private Network

WAYF Where Are You From

WSDL Web Services Description Language

XML Extensible Markup Language

1

Introduction

Contents

1.1 Motivation

The Secure idenTity acrOss boRders linKed (STORK) project[1] aims at creating a single digital identification and authentication area over Europe, establishing electronic Identities (eID's) for individuals and legal entities.

To achieve this goal, the interoperability of electronic Identity Management Systems (IdMs) across the European Member States (MSs) is crucial. Countries like Austria [7–10], Estonia [11, 12], Finland [13], The Netherlands and others already have IdMs in place that were implemented without the establishment of clear common goals across the European Union (EU). This has caused a high heterogeneity of implementations which makes interoperability difficult and poses several security and privacy challenges.

Some MSs also deployed SmartCards (SCs) for their citizens [14] which function as electronic identity cards. These cards are called European Citizen Card (ECC), and have built-in support for the Public Key Infrastructure (PKI) enabling Electronic Identities (eIDs). This means that these eID cards provide the necessary functionalities to citizens that allows them to authenticate with high assurance to online services. However, there are countries (e.g. the United Kingdom) that still don't provide any type of Identity Card implementation at all.

Because of the different IdM solutions already adopted by those MSs, a citizen within the EU might have multiple security tokens (such as SCs or Subscriber Identity Module (SIM) cards) that store his identity, credentials, mandates and often provide a mechanism for strong (i.e. that has a high level of assurance) authentication. This further increases the interoperability problem as the system must provide compatibility with already existing solutions and must be flexible enough to accommodate the integration of possibly new eID schemes.

The creation of such eID schemes also introduces the possibility of more secure and usable authentication systems that provide Single Sign On (SSO) [15]. Everyday we face the problem of having multiple Virtual Identities (VIDs) scattered across different Service Providers (SPs), which causes us to have to manage and remember a wide range of credentials and often leads to poor and unsecure password choices. SSO and IdM solutions are usually used within organizations to minimize this problem, but with the ever growing number of services on the web, this is not sufficient. The demand for a national and European interoperable standardization of such systems is becoming a necessity [16].

Even within a MS, citizens attributes are often scattered across multiple IdMs. One might be a student at a university, which stores his academic credentials, but might also have attributes stored by the government, such as information about the releasing of his drivers license or his current employment status, etc... Every time the citizen necessitates to give proof of multiple attributes scattered across different IdMs, he would currently have to individually gather all of the different attributes and present them to the requesting entity. To solve this problem, an attribute aggregation system [17] is required, so that a user can do this automatically using his eID. This arises other problems such as: legal issues; assuring the user privacy across the different IdMs transactions; give the user information about the policies of release of his attributes; enforcing these policies at the SPs; and allow the user to maintain his anonymity (to an extent). Moreover, the user shouldn't need to authenticate to all IdMs individually, but instead, a relation of trust should exist between organizations to provide SSO across them, improving usability and security.

The privacy of European citizens is a fundamental right, and for the project to succeed, a reasonable level of assurance must be given in order to let them build trust in the system and let them rest assured that their data is securely protected and safely delivered across MSs. The residual risk of using the system will only be accepted by citizens and organizations if a certain level of trust, assurance and

[1] https://www.eid-stork2.eu

usability is achieved, enabling the uptake of the system.

1.2 Problem Description

By creating a single European electronic identification and authentication area the STORK project enables the passage and validation of citizen's attributes through different MSs. The deployment of such infrastructure creates the foundations for the flourishing of new web application and services that citizens can use and access by presenting strong authentication credentials and digital attributes. The grand goal of STORK is not only to facilitate collaboration and free exchange of these attributes within the EU, but also the elimination of the existing dependency between bureaucracy and paper signatures. Even though in many countries governments currently store their citizens identity data digitally and cryptography provides us the necessary tools to verify digital data integrity, authenticity and guarantee non-repudiation, it's still the case that for the release of documents containing sensitive data, government bureaus rely mostly on face-to-face authentication (by verifying the information on some citizen's government emitted document) and its verifiability depends on stamps and/or signatures on said document. The current state of affairs is inefficient, cumbersome for both governments and users and has several limitations. Moreover, It has been proven that cryptographic signatures provide higher security than their handwritten counterparts by being much more difficult to forge.

The scope of the STORK project is very vast and branches to different domains: eBanking, eHealth, Public services for Business and eLearning and Academic Qualifications. In this work we will describe the implementation of the latter: the STORK 2.0 Portuguese eLearning and Academic Qualifications pilot, which is composed by the following systems:

- The Attribute Provider (AP) of IST, which is to serve STORK academic related complex and simple attributes, upon requests received from the Portuguese IdM.
- The Trusted Attribute Display Service (TADS) Service Provider for the materialization and dematerialization of digitally signed documents across European MSs.
- The eLearning pilot, in which a student can attend an online course from a participating university by authenticating through the STORK infrastructure and presenting the necessary attribute to prove his enrolment.

As part of the academic Large Scale Pilot (LSP), the three systems mentioned above create the necessary testbed for the academic pilot within Portugal, enabling cross-border testing with other participating countries for authentication and the retrieval of attributes between them.

The main challenges of this work are the development of the eLearning and AP components of the pilot and their integration with the university information system and national IdM, and also, the development of the stand-alone TADS application that will be connected to the STORK infrastructure, and whose goal is to give its users the capability of retrieving STORK attributes of their choosing from several sources (APs) and aggregate them into a printable document, verifiable through its digital signature. Because printing the document makes it lose its digital signature, third parties will be able to link the printed document to the original digital version, making it possible for them to verify its veracity. This service is to bridge the current existing dependency on paper documents and their digital counterparts, however, the purpose of TADS is not to perpetuate the usage of paper documents, but instead, to incentive the deployment of APs that can be used directly by SPs through STORK and in the long run dismiss the usage of paper altogether, by demonstrating that the infrastructure facilitates the integration of these two entities.

1.3 Objectives

The main objectives of the STORK 2.0 Academic and Qualifications pilot components are the following:

- Develop and test an AP for academic attributes at Instituto Superior Técnico (IST) information system and integrate it with the national IdM and hence, to the STORK infrastructure.
- Develop an eLearning platform (i.e. a SP) within IST information system and integrate it with the STORK infrastructure to test cross-border academic attribute retrieval for the online course access control.
- Develop the TADS and demonstrate the viability of the dematerialization/materialization of simple and complex attributes across the established STORK trust web between MSs.

The several components of the pilot have different key roles in the above objectives, but the final goal is that they lead to the end users being able to try the two offered services (eLearning and TADS) to test the cross border attribute retrieval, demonstrating the viability of the project as a whole, as well as the security and privacy measures that have been taken. This should build the trust of the users in the project and show them its potential for EU citizens (specifically EU students for this pilot), generating awareness and hopefully leading to the uptake of the infrastructure by SPs, EU governments and citizens.

1.4 Contributions

As part of the STORK 2.0 Academia pilot, this work contributes with the development of the TADS web application to be deployed at IST and several other academic institutions participating in the LSP. TADS has been designed with usability and ease of configuration and deployment in mind.

Since it wouldn't make sense to deploy TADS nationally without having an AP capable of serving simple and complex attributes (such as the European Diploma Supplement (EDS)) for national students, the other contribution that derives from this work is the development of the AP integrated with the Portuguese IdM system (managed by Agência para a Modernização Administrativa (AMA)[2]) and IST's academic information system (Fenixedu-academic)[3] that will serve academic related attributes.

Beside the AP, the institution's academic information system will also be hosting the eLearning pilot. This is a service that foreign students will be able to access by enrolling in the offered eLearning course at their own institution. To access the course, the student will be prompted to authenticate through the STORK infrastructure and retrieve the necessary attribute that prove his enrolment, granting him access at the hosting university information system. This leads to another contribution of this work, which is the integration of the eLearning pilot into IST academic information system Fenixedu-academic and Central Authentication Service (CAS) server. Moreover, STORK authentication will be offered not only to foreign students trying to access the eLearning course of the LSP, but also IST students, offering a stronger authentication alternative using the Portuguese IdM and Citizen's Card.

Finally, we co-wrote a paper to be submitted to a scientific journal in collaboration with researchers from other participating countries of the project, however it's still being worked on, so it hasn't been submitted yet.

[2] http://www.ama.pt/
[3] https://github.com/FenixEdu/fenixedu-academic

1.5 Solution Outline

An overview of the infrastructure and how the different STORK 2.0 components of this pilot interact with each other is displayed on Figure 1.1. The components and services follow a Representational State Transfer (REST) architecture and communicate using the Security Assertion Markup Language (SAML) standard. Because of its user-centric design, most interactions imply redirecting the user browser to the next involved component which will present the user with a set of options to give him control over the retrieval and release of his attributes.

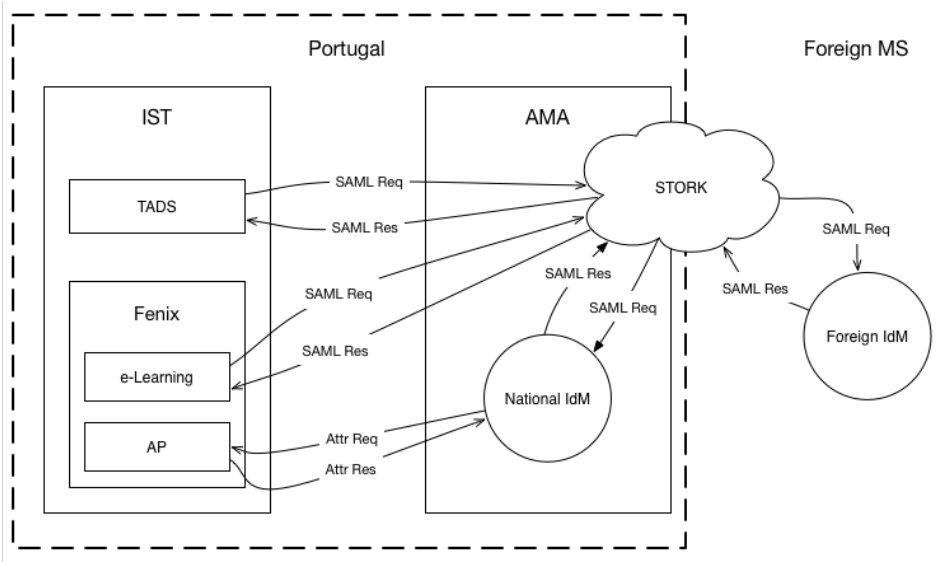


Figure 1.1: Macro overview of the STORK architecture and its integration with the national IdM.

As seen in Figure 1.1, both the TADS and eLearning components act as SPs hosted within IST, where the former is a stand-alone web application, and the latter is the combination of two other components: the CAS service, responsible for authentication and access control to the second component: the eLearning course hosted on the Fenixedu-academic (which we will reference to as Fenix for short) information system. SPs request STORK authentication and optionally attributes creating a SAML request that is then sent RESTfully to a network of proxies called Pan-European Proxy Service (PEPS). This network routes the requests to the correct Identity Provider (IdP) of the citizen's MS so that an assertion regarding his identity can be retrieved. In Figure 1.1 can be observed that either TADS or the eLearning applications can initialize the STORK authentication process sending the request to the infrastructure. The SAML request is then processed and redirected to the national or foreign IdM, depending on the nationality of the user (in the figure both flows are represented). After the user is successfully authenticated at his IdM (if attributes were also requested they are also retrieved), and the SAML response is created and sent back through the infrastructure. An alternative to the proxy network exists via the STORK middleware (not represented in the figure). We will elaborate further on these matters in section 2.3.12.

Next, we'll briefly describe the solution outline for the individual components.

1.5.1 The Trusted Attribute Display Service (TADS)

The TADS is a web application built using Node.js, a JavaScript framework. The Node community offers a wide array of open-source modules that provide server-side and client-side functionality. The framework also includes a built in HTTP and HTTPS server. These feature make it very easy to manage

dependencies, prototype, test and deploy web applications.

TADS is hosted within the informatics services of IST and is publicly accessible[4] with any modern web browser. The web application presents the user with three options: one to create a document, one to manage documents the user might have previously created and a service for the verifiers (someone trying to verify a document created with TADS) to validate a printed document. The created documents containing the attribute names and values are digitally signed using a SignServer[5] instance and are stored in a noSQL mongoDB[6] database. The user can access or delete these documents at will by accessing the service and authenticating through STORK. When created, the documents are also assigned a QR Code that contains an unique identifier. This identifier's purpose is to facilitate the validation of the document by the verifier. To do so, the verifier must access the TADS portal and go to the document verification service, where after e-mail validation, will be directed to a web page that allows to scan the printed document QR Code with his computer's webcam (an alternative is provided if the user doesn't wish to give the browser's access to the webcam, or simply doesn't have one). This allows him to retrieve the original, digitally signed Portable Document Format (PDF).

1.5.2 The Attribute Provider

The developed AP is a Simple Object Access Protocol (SOAP) web service integrated into the Fenixedu-Academic platform and compliant to a Web Services Description Language (WSDL) file supplied by AMA. All national APs must abide by this WSDL specified by the Plataforma de Interoperabilidade da Administracao Publica (PIAP), a component of the national IdM. Because all information regarding any student or staff members of IST (such as curriculum and roles, for example) are stored by the academic information system, the AP has to be integrated into it to be able to access the relevant information necessary to respond to the requested attributes. The SOAP request that the AP receives at its endpoint only contains the civil identification number of the requestor, and the constructed response must contain all the academic attributes that exist for the given user, if any.

1.5.3 The eLearning Platform

The STORK 2.0 elearning pilot at IST will also be deployed and integrated into the Fenixedu-academic information system, but will also require integration with IST's CAS deployment. The system will allow a foreign student to access an eLearning online course on the platform, by authenticating himself on the IdP of his country of origin and obtaining the necessary credentials to attend the course using the STORK infrastructure. The course will be hosted on Fenix because the platform already provides the necessary tools for an online course creation and content management. The CAS server requires integration as it is the authentication portal deployed at IST, and it controls all students authentication and access to Fenixedu-academic. Hence, the CAS server must be able to consume and validate incoming STORK SAML Responses with the student's authentication information and the attributes that allow his on-demand enrolment and access to the course. Besides the eLearning students access, CAS will also be prepared to allow IST students to use their Portuguese citizen's card for authentication through the STORK infrastructure using the national IdM, alongside the normal username/password login option.

[4] tads.tecnico.ulisboa.pt
[5] http://signserver.org/
[6] http://www.mongodb.org/

1.6 Thesis Outline

The remainder of this document is structured in the following manner. Chapter 2 we describe the current state of the art in IdMs and relevant concepts to better understand this work. Chapter 3 introduces some of the technologies utilised for the development of the solution. Chapter 4 explains and illustrates the proposed solution in detail. Chapter 5 analyses the solution from a security perspective. Chapter 6 completes this work with the drawn conclusion and possible future work. Appendix A illustrates the interfaces of TADS and CAS with some screenshots. Appendix B shows the results of a security assessment of the deployed HTTPS protocol for TADS. Appendix C provides examples of SAML messages. Appendix D contains an example of a document created with TADS containing some attributes retrieved from STORK.

2

Related Work

Contents

This chapter will present the relevant literature for this work.

To understand the operation of the STORK LSP and its implications, it is important to discuss the current state of the art of IdMs and related technologies. We will start by reviewing the most commonly used security tokens for identification and authentication within IdMs. We will then move on to some concrete IdMs designs, architectures and implementations, followed by authentication systems and finally, federated IdMs, whose scope usually covers the former two technologies.

2.1 Concepts and definitions

To better understand this work and the related bibliography we will now introduce some important definitions.

2.1.1 Digital identity

What is digital identity? The definition is not unique, but despite it having several interpretations, it falls under the common ground of being a set of attributes that represent a person, given a specific context or role (i.e.: online service user, an institution's student, company employee, etc. . .). It is important to note that a digital identity may or may not have certified information about the user, differentiating it from the person's real identity [18]. Most digital identities are composed of three different types of data:

- Identifiers: A unique piece of data (a series of digits, a string or any other) that uniquely identify a subject. Some examples: URIs, passport, employee or student numbers.
- Attributes: Data that contains information about some characteristic of the subject. Some examples are: given name(s), surname(s), employer, title(s), gender, age, roles and address.
- Credentials: A set of data that proves one or more claims about an identity. The reliability of credentials varies depending on the technology used, some examples are: passwords, Kerberos tickets, PKI certificates and SAML assertions.

Because digital identities are often contextual to a specific domain, a person will usually have several identities scattered across domains with different purposes (e.g: one for work, one for university and another at an online forum or retailer). These different identities often become burdensome for the user because they have different attributes, assurance levels and credentials he has to manage.

2.1.2 Federated Digital Identity

An identity federation is an established partnership between organizations, countries or any number of entities to allow the exchange of individuals identities information and attributes using common agreed namespaces, protocols and technologies to enable interoperability and create a network of trusted relationships between its members [19]. Because federations involve multiple organizations, problems such as liability, trust and security become much more relevant [20], as many more stakeholders are involved in a federation, making federated digital identity management much more challenging than managing eIDs within a single organization (at a single-domain level).

2.1.3 Identity Management Systems (IdMs)

Now that we have a definition for digital identity, we can define IdMs as the systems that aggregate technologies and standards that enable the process of maintaining and securing the integrity of digital

identities, allowing services to access their associated data (attributes and credentials) securely while keeping the privacy of their respective owners [21].

IdMs usually involve the following entities:

- Identity Providers (IdPs): provision assertions about a person's identity. In most IdMs, this is the entity where the user can authenticate himself using his credentials. Upon successful authentication, the identity assertion is returned by the IdP, so that the user can give proof of his digital identity to Relying Parties. It is important to note that IdPs and Attribute Providers are sometimes the same entity (this will be better explained in subsection 2.1.4).
- Attribute Providers (APs): are entities that can provide attributes about a certain identity. APs normally store identity attributes that are relevant in a given context, for example, a company stores information about the identity of its employees and their roles. A university, on the other hand, will store information about its students and their academic curriculums. In both cases the institutions can provide accredited information, the former about the person's current role in the company and the latter his qualification and degrees. Another example would be a government citizen registry, that can provide information about a number of citizen's attributes, such as his current address and date of birth, for example.
- Relying Parties (RPs): (also commonly referred to as Service Providers (SPs) in the literature) are the entities that rely in the identity information provided by IdPs and attributes released by APs to infer access control privileges and/or offer customized services to the user. Examples of this are governments offering bureaucratic services online, with the requirement that the user presents an identity assertion that can give high assurance about the truthfulness of such claim, or some online services that require the user to prove that he is at least eighteen years of age to access its contents.

2.1.4 Identity Management Models

IdM models can be divided into four categories: isolated, centralized, federated and decentralized (distributed) [22–24].

2.1.4.A Isolated Model

In the isolated model, the IdP and the SPs are the same entity. This is the most common model currently on the Internet, where SPs are responsible for the management of their users credentials. Even though this model is the most popular, it is by far the one with the most drawbacks: it is very burdensome for the users, which need to remember their credentials for each SP, often leading to poor/insecure password choices, hence lowering security; it is also burdensome for the SPs themselves, as managing user credentials has infrastructure and security costs; and finally, SPs control over credentials might disclose the user private information without consent, leading to privacy loss. SPs often claim to protect the users privacy through Privacy Policy Agreements (PPAs), but to what extent they are enforced is often a grey area.

2.1.4.B Centralized Model

The centralized model separates the IdP and SPs roles, having a single centralized IdP that stores all the users credentials/attributes. This is often used in mid-to-large organizations where SSO solutions are in place, alleviating some of the burden of the users having to manage multiple credentials, as they

will use the same set across all the services provided within the organization. It also alleviates the management of credentials on the organization itself, as it eases the management of high numbers of users by having a single place of storage. This model however, suffers even more of the privacy issue, as more data of the user is stored in a single place; which is also a security threat as the centralized IdP is a single point of failure.

2.1.4.C Federated Model

The federated model establishes an identity federation where SPs rely on any IdP within it. By relying on the trust network established, the SP might even not previously know the IdP where a user authenticates. This has many advantages: SPs need not store any user information, enhancing privacy and being more cost-effective. It provides SSO functionality across different SPs, enhancing usability and overall security for the user. A drawback of this model is the introduction of the IdP discovery problem. Another is that since the user information might be scattered across different IdPs, a solution for attribute aggregation must be implemented. These drawbacks introduce additional complexity to the system but alleviate the SPs from identity management altogether, having to exclusively manage the federation trust network (which usually involves PKI Public Key Certificates (see section 3.1)).

2.1.4.D Distributed Model

In the distributed model, the IdP is a security token (such as a SmartCard, SIM card or any other) held by the user itself. The credentials are securely stored in the token and are released upon user authentication through a Personal Identification Number (PIN) upon request. These tokens are issued by a trustworthy entity (such as the government or a national bank), which validate the trustworthiness of the stored credentials. This is the model that gives more control to the user over his credentials because he is in possession of the tokens. A disadvantage however, is that SPs need to handle a possible wide range of different tokens and the inherent technologies and standards. The tokens also have other problems: they usually have little storage capacity (e.g: SmartCards) and also, the secure update of outdated credentials within the tokens is difficult. These solutions have to rely on distributed APs (the tokens) and additionally have to face discovery and attribute aggregation problems.

2.2 Fundamental Technologies

In this section we will briefly introduce the most relevant technologies that are part of or constitute most IdMs.

2.2.1 Security Tokens

Some IdMs national infrastructures (such as Austria [8]), allow the use of tokens issued by private or public sector organizations as an eID, as long as it supports qualified electronic signatures. This is a technological neutral approach, which allows the users to have several eID tokens at their disposal. A wide array of different technologies can be used for the tokens (e.g.: mobile devices, SmartCards and others).

2.2.1.A Subscriber Identity Module (SIM) cards

These small chips are essentially a smaller version of a SmartCard (integrated circuits in a pocket-sized card) widely deployed in mobile devices, as they are used by mobile carriers to identify and au-

thenticate its users through the built-in cryptographic functionalities. Using mobile carriers as trusted IdPs and an IdM system, mobile devices could become viable identity tokens [25].

2.2.1.B European eID citizen cards

A number of European MS have implemented and adopted the European Citizen Card (ECC), a SmartCard with cryptographic features that provides three functionalities: identification, authentication and a cryptographic signatures. To provide these functionalities, the ECC relies on the PKI and stores one or more certificates signed by a trusted Certificate Authority (CA). The ECC is a credential system with high accountability and a strong authentication level (it authenticates its holder with high assurance) [26].

2.2.1.C Mobile Transaction Authentication Number (mTAN)

The mTAN token depends on the used mobile phone technology and creates a one-time password to authenticate a user or to confirm a transaction. When a user begins some identity dependant operation (an authentication or a bank account transfer for example) a server-side one-time password is generated and is sent to the user mobile phone through an Short Message Service (SMS), which he will then insert into the browser to confirm the transaction. The SMS contains information about the transaction so that the user can verify that the details the service received are correct. This is considered a stronger authentication method because it is unlikely that an attacker has access to both the user mobile phone and the user authentication credentials. However, with the ever increasingly number of smartphones connected to the internet nowadays, the user might use his device to login to the service and receive the mTAN, discrediting the former premise [27].

2.2.1.D Authentication Grid Cards

These cards hold a matrix with random characters in each of its positions. The cards are issued to the user by the SP and are delivered using a communication channel that differs from the one used by the service itself. When the user wants to confirm a transaction after he logged in into the service, he is prompted with one or more characters from the matrix. Just as in the mTAN, this method is also considered secure because it seems unlikely that an attacker possesses both the user credentials and the security grid card, as they are delivered through different channels.

2.2.2 Single Sign-On

Single Sign-On is the use of the same authentication credentials on different systems or services. In most cases, SSO solutions are deployed within the same institution or organization, but with the growing support for *federated identities*, SSO across different organizations (also called multi-domain SSO) is becoming more common. The main feature of SSO is that a user won't have to authenticate multiple times to access different services. After he authenticates the first time within the current browser session, he won't be prompted again for the credentials upon access to services that are provided within the network of trust of the federation.

2.2.3 Representational State Transfer

Representational State Transfer (REST) [28] is a scalable web application architecture where resources are identified with Unique Resource Identifiers (URIs) and can be retrieved using the HTTP

protocol and its respective verbs (GET, PUT, POST, DELETE) for the communications between client and server. The resources are usually retrieved as HTML, XML or JSON which makes the resources easy to access with any browser. The architecture model imposes the constraint that the server must be stateless and it's up to the client to store all the context data, sending it to the server with the requests, making the web applications cacheable and scalable. REST applications became popular with the rise of the Web 2.0, where user customised services and user generate content became the norm.

2.2.4 Security Assertion Markup Language

SAML is an XML-based open-standard designed for federated authentication, authorization and attribute exchange, maintained by the OASIS[1] global consortium [1].

A SAML interaction involves, at a minimum, two entities: the SAML *asserting party* and the *relying party* (or SP). The *asserting party* is any entity that creates SAML assertions, while the *relying party* is an entity that consumes the assertions it receives. If one of these entities makes an explicit request to another, it is called a requesting party (or *SAML requester*), and the responding party the responder (or *SAML responder*). A *relying party's* willingness to trust an *asserting party* depends on the previous existence of a trust relation (established through the PKI) between the two entities.

SAML assertions usually refer to some *subject* (a security principal - an entity that can be authenticated), about whom something is being asserted. An example could be that *subject* "John Doe" has the title of "Engineer". The *relying party* that receives such SAML assertion may then decide to grant "John Doe" web SSO and access to some local resources depending on its access control policies.

Examples of a SAML Request and Response messages (issued and received by TADS) are provided in Annex C.

SAML Components

The SAML standard is composed by four major concepts: Assertions, Protocols, Bindings and Profiles. These concepts form the building blocks that when put together allow the standard to adapt to a number of different use cases. Figure 2.1 shows an overview of these concepts and how they relate.

Assertions are statements about a *subject* that may contain authentication, identity, entitlement information or attributes. Assertions are of the following types:

- Authentication statements: Created by the IdP where the user authenticated by presenting valid credentials;
- Attribute statements: These contain specific information regarding the subject (e.g.: date of birth, name, surname, etc. . .);
- Authorization decision statements: These define something that the subject is entitled to do (e.g.: access some premium features of a service subscription).

Protocols define a number of generalized request/response messages that specify how the SAML Requests and Responses should be formed depending on the use case. Some of the most relevant protocols are:

- Authentication Request Protocol: Defines how a principal can request an authentication assertion and optionally attribute statements about the subject.

[1] https://www.oasis-open.org/

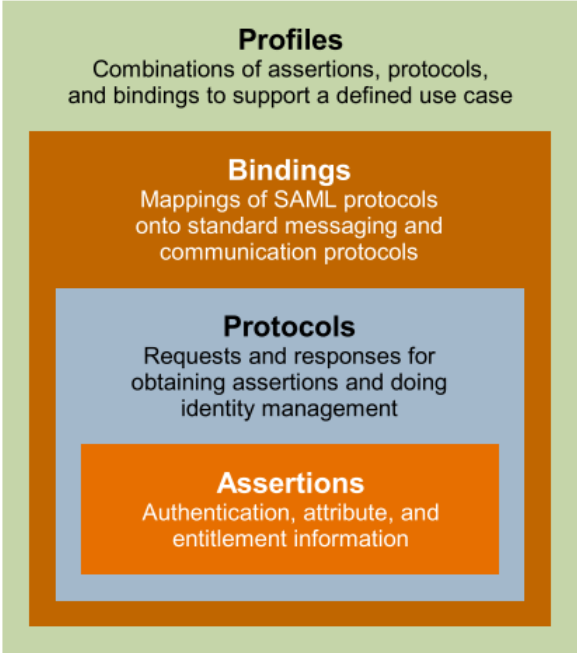


Figure 2.1: SAML basic concepts [1].

- Single Logout Protocol: Defines a mechanism to logout of active sessions on multiple federation associated with a principal.
- Assertion Query and Request Protocol: Defines a set of queries to obtain SAML assertions. These queries can be used by an SP to ask for assertions it requires.

Bindings specify how a SAML message is carried over the underlying transport protocols that are supported. Some examples of bindings are:

- HTTP Redirect Binding: Defines how to transport SAML messages using the HTTP redirect messages (HTTP status 302 or 303).
- HTTP POST Binding: Defines how to transport SAML messages on the HTTP headers as a base64-encoded URL parameter.
- SAML SOAP Binding: Defines how SAML protocol messages can be transported within SOAP messages.
- SAML URI Binding: Defines a means to retrieve an existing SAML assertion by resolving an URI.

Profiles define how the components stated above combine together to be used in specific use cases. Some examples of profiles are:

- Web Browser SSO Profile: Defines how SAML can request authentication and attribute assertions to achieve multi-domain (federated) SSO.
- Identity Provider Discovery Profile: Allows SPs to discover IdPs a user has previously visited.
- Single Logout Profile: Defines a mechanism for SPs to Single logout using a number of different protocols.

We will focus mainly on the Web Browser SSO Profile, which is the most commonly used profile (and the one adopted by STORK). It will be explained further below.

Web Browser SSO Profile

The Web Browser SSO Profile provides a variety of options regarding two primary aspects: (1) who initiates the flow (SP or IdP) and (2) what bindings and protocols are used to deliver the messages between them.

Choice nº (1) depends on where the user starts the process for the web SSO. The most common is the initiation at the SP. This happens when the user is trying to access some resources at the SP website and is not logged in. Hence, before granting access to the resources, the SP has to send the user to an IdP to authenticate. The IdP then creates an authentication assertion and sends the user back to the SP. The SP then processes the assertion and determines whether to grant the user access to the resource. The other situation is when the flow is started at the IdP. This can happen, for example, if the user is already logged in at the IdP, and wishes to follow some link to an external resource provided by a SP. In this case, the IdP simply creates a SAML authentication assertion and redirects the user's browser to the SP's SAML consuming endpoint. The SP receives the SAML assertion at the endpoint, processes it and decides whether to grant access to the resource or not.

Choice nº (2) decides which SAML bindings are used to transmit the SAML messages between the SP and the IdP. The bindings are interchangeable, meaning that in different communication directions different bindings may be used. For example, the SP may use an HTTP binding to send an authentication request, and the IdP may respond with a SOAP binding. The Web SSO profile allows three types of bindings for an authentication flow started by a SP:

- HTTP Redirect Binding: The SP sends an HTTP redirect response to the browser (HTTP status code 303), with the IdP authentication URI in the HTTP header along with a SAML AuthnRequest message encoded as a URL query variable named SAMLRequest. The browser processes the redirect and issues an HTTP GET to the provided URI with the SAMLRequest query parameter.
- HTTP POST Binding: Very similar to the Redirect Binding, but instead of an HTTP redirect, a POST to the provided URI is issued, and the SAMLRequest (or SAMLResponse) is placed within an HTML FORM as an hidden form control named SAMLRequest (or SAMLResponse).
- HTTP Artefact Binding: Instead of delivering the SAML response in the HTTP header as a parameter, the browser is redirected to the SP's assertion consumer service endpoint with an artefact. The SP is then responsible to communicate with the IdP Artefact Resolution Service (a SOAP protocol) to retrieve the SAML Response associated to that artefact.

The typical Web Browser SSO Profile interaction involves three entities: the User, the IdP and the SP. An interaction example would be as follows:

1. A user wishes to access some service resource following a link on his web browser;
2. The SP requests that the user authenticates to check if he is eligible to access the resource depending on its access control policies;
3. The SP creates a SAML authentication request and redirects the browser with the HTTP Redirect Binding to the IdP's authentication URI with the SAML request in the HTTP header as a query parameter called "SAMLRequest";
4. The browser processes the redirect and issues an HTTP GET with the SAML request in the URL parameters;
5. The IdP receives and processes the SAML request, and returns to the browser the authentication page;

6. The user inserts his credentials and submits them;

7. The IdP verifies the credentials, and if valid, issues a SAML response with the an Authentication Assertion that proves the user has successfully authenticated. Optionally, the IdP might also insert Attribute Statements that might have been requested by the SP;

8. Finally, the IdP issues an HTTP POST to the browser with the SAML response as a hidden HTML FORM control called "SAMLResponse" back to the assertion consumer service URI of the SP;

9. The SP receives the SAML response and verifies it by checking the Authentication and Attribute Assertion's cryptographic signatures, extracts the authentication principle and attribute values and decides whether to grant access to the resource or not based on its access policies.

Persistent or Transient Pseudonym Identifiers

Even in federated SSO, the privacy of the user might be a requirement or of high concern. For this reason, the SAML standard supports either transient or persistent identifiers that can be used to protect the user's identity between the parties involved in SSO and/or attribute exchange.

To do so, the federated identity of the user must be linked to the local login accounts he has at the different members of the federation. This process is called *account linking*, and once it's done the federated identity will be used to represent the user across the different domains he accesses, independently of the identifiers that are used locally. Once a pseudo-random identifier is used to identify the user instead of his login username, the entities that are part of the federation are able to link the accounts, enabling them to provide SSO without disclosing and correlating the local identities of the user over the different domains, hence protecting his privacy. When *account linking* is done, **persistent pseudonyms** are used, as the entities need to store a link between the local account and the federated identity.

An alternative to *account linking* is the usage of **transient pseudonyms**. With transient pseudonyms, the SP never stores any information about the user and doesn't maintain any local login account. Instead, the SP relies on some SAML assertion only to grant access to some resource based on a credential the user presents or simply by presenting the authentication assertion, and can cache the pseudonym temporarily to enable SSO. This way, true anonymity is achieved, as the SAML assertions contain no information identifying the user, and the SP has no other data about either the digital or real identity of the subject.

An example of the usage os persistent pseudonyms is as follows:

1. A user tries to access a resource on *trains.example.com*, but he has no current login session on the website. The resource that the user was trying to access is saved in a SAML field called *RelayState* (this is to save SP specific metadata that may be useful once it receives the SAML Response);

2. The website creates a SAML authentication request with the *RelayState* data and requesting that the IdP provides an assertion using a persistent pseudonym for the user (this is done setting the SAML field *NameIdPolicy* to true). The SP then uses the HTTP Redirect Binding to redirect the browser to the IdP (*travel.example.com*) along with the SAML request;

3. The user is challenged to present his login credentials for his account at the website *travel.example.com*;

4. The user provides his credentials, logs in and is locally identified as user "john" (his principal for the account);

5. Seeing that the SAML request has the field *NameIdPolicy* set to true, and that the request explicitly demands the usage of a pseudonym, the website *travel.example.com* creates a pseudo-random string (e.g: 762534) as a pseudonym for the user and associates it to his local account. The SAML web SSO assertion is created with the subject's value set to the newly created transient name identifier. Optionally, other attribute assertions are also added to the SAML message (e.g: the user's membership level);

6. *travel.example.com* issues an HTTP POST request to send the SAML response back to the SP's assertion consumer service;

7. *trains.example.com* assertion consumer service processes the SAML response and verifies its validity. The supplied pseudonym is then used to verify if a previous federation already exists for that identifier. If it already exists, it means that *account linking* has previously been established - go to step 9. Otherwise, the local account still has no pseudonym linked. In this case, the SP will ask the user to authenticate with his local account credentials;

8. The user provides his credentials for the local account (with identifier "joe") and the pseudonym is stored along with the name of the IdP that provided the pseudonym. The federation between the two accounts is now established;

9. A local login session is established at *trains.example.com* for the user "joe" and an access check is made to verify if the user has the necessary authorization to access the resource (obtained from the *RelayState* field of the SAML response);

10. If the user has authorization, access is granted and the requested resource is returned to the browser.

For transient identifiers, the example is very similar, however, at step 7 the SP doesn't create any *account linking* (as no local account exists for the user), but instead, it only checks if the user has access to the resource he requested (either by presenting the authentication assertion being sufficient or using some additional attribute assertions to make a decision about access control). If access is granted, the transient identifier is cached so that it is valid for the current established session, and the resource is returned to the user.

The Name Identifier Management Protocol defines a way to terminate an established federation. This allows a SP or an user to initiate a protocol that will basically ask the entities with an established federation to revoke it and delete the persistent identifier linking the local account to the federated identity. The initiating entity will send a message to the other containing the pseudonym and asking the other to remove it. If the requested entity responds with an affirmative response the initiator also deletes its entry, terminating the federation between the two accounts of the user.

2.3 Identity Management Systems

This section will describe the state of the art in Identity Management Systems.

2.3.1 Shibboleth

Shibboleth is an open-source IdM system that provides SSO functionality with federated identity support [29]. The system has two main software components: the Shibboleth Identity Provider (SIdP) and the Shibboleth Service Provider (SSP). The first is where user attributes are stored and the second

is the software running on the SP that wants to secure its resources. When a user navigates to a SP that uses the SSP, he is redirected to a page called Where Are You From (WAYF), where he is prompted with a list of organizations he might be associated with. After choosing an organization, the browser is redirected to the respective login page. The user is now seeing the familiar login page of his organization and enters his credentials. After authentication occurs, the SIdP sends the browser back to the SP website and sends an assertion to the SSP that such user is logged in. After the SSP verifies the assertion and its validity, it asks the SIdP where the user logged in for any additional attributes it may need (e.g: "organization role", "identifier"...). The SSP can now use these attributes to decide if should grant access to the user or not. If the organization authentication method is SSO and the user already has a session established, the next time he tries to access a resource the login web page won't be displayed, making the login process transparent to the user.

The Shibboleth System uses the standard SAML 2.0 (see section 2.2.4) for its assertions, so that Federated Identity can be achieved across multiple systems that support the standard.

One of the strongest points of Shibboleth is that it gives high granularity control to the users over the release policies of their stored attributes. Besides this, an user can get access to a SP without releasing any attributes (unless the SP explicitly requires some).

An effort is being made to create a group of federations that use Shibboleth to constitute a trusted network of academic attributes.

2.3.2 Microsoft CardSpace

A single user can have multiple digital identities over a multitude of different IdPs, which makes it difficult to have an aggregated, global view of all the online services he accesses, making their management difficult because the user can't have a global view of all his identities over the online services he accesses.

Microsoft's CardSpace is a solution based on a software client that allows the user to store metadata and manage his different identities through a simple interface [30]. This interface tries to make an analogy to the user wallet, by displaying several cards (called infocards) which represent his identities, just like the different cards in our wallet usually assert our citizenship, institution memberships or other attributes about us.

The different identities (or assertions) that a user has are managed by the Microsoft CardSpace Metasystem (MCM) by storing metadata about the different VIDs of the user.

We will now describe a user interaction with the system: the user starts by requesting an identity based service (step 1). If the service is being accessed through a browser, the user will be redirected to the authentication page where an option to login with Microsoft CardSpace is available. If the user chooses this option, the MCM will ask the service provider for a token policy (step 2). The SP will then respond to this request with his token requirements and security policies (the SP basically specifies the claims he is able and willing to accept) (step 3). The SP also gives the endpoint address of the IdP for the required security token (if not, the user may be prompted to provide the URL for the IdP to be used). Moreover, the SP might propose the type of *proof key* in the issued token. This *proof key* is used to establish that the token was issued on behalf of the user, and by proof of possession, that the user is in-fact authorized to use it. Depending on the SP security policies, a certain *freshness* for the token may also be requested to ensure protection against Replay Attacks. All the established connections enforce the use of SSL/TLS to guarantee confidentiality. At the end of these three phases, the MCM now knows what claims the SP is asking for. On the MCM side, the user will now be displayed the SP identity, which he may choose to accept or not (step 4). In case of a positive response (step 5), the MCM will decide what infocards can be used based on the security policies received in step 3, and highlight them for

the user to decide if he is willing to release their respective attributes or not (steps 6, 7 and 8). After the user selects the infocard, the MCM will contact the IdP to ask what kind of authentication policies are required by the user (step 9). Depending on the IdP security policies, stronger or weaker types of authentication may be requested (username/password pairs, Kerberos tickets, self-issued tokens or PKI certificates with key pairs stored either via software, or some security token such as a SmartCard) (step 10). After the IdP specifies the authentication process, the MCM will prompt the user to authenticate itself (step 11). Once the user inserts his credentials and confirms (step 12), these are sent (step 13) securely to the IdP and after the user successfully authenticates, the security token is sent to the MCM (step 14). The MCM shows the obtained token to the user (step 15) and if he is satisfied with it (step 16), it can now be used to access the requested service from the SP. If the security token is not signed by the IdP it was requested from, the user might optionally sign it with the public key of the SP that will consume it (this is optional step 17). Now, the MCM can send the token to the SP (step 18), which, after proper validation, will respond by granting service access.

Just as in Shibboleth (see sec. 2.3.1), Microsoft Cardspace doesn't allow the aggregation of multiple attributes into a single infocard. This is a problem, because a SP might request attributes that are stored on different IdPs, rendering the system unusable in such situations.

2.3.3 Trusted Attribute Aggregation Service (TAAS)

TAAS is a web based federated identity management service that uses the SAML 2.0 standard and is loosely based on Microsoft CardSpace [2, 31].

With the growing number of IdM systems, it is unlikely that a user ends up having all of his attributes stored in just one of them. For this reason, an aggregation service is needed. TAAS allows to aggregate a user attributes from multiple sources with only needing to authenticate at a single IdM and keeping his privacy across retrievals. Furthermore, the user has fine grained control over the release policies of his stored attributes.

For this model to work, it is assumed that some IdPs and SPs have pre-existing trust relationships, and that a SP trusts all IdPs from which it receives attributes from.

Model

To use TAAS, users navigate to its web portal and authenticate using an IdP that has a trust relationship with the system (this is similar to Shibboleth's approach). After the user is authenticated, the IdP will show him a list of attributes that he may release to TAAS for aggregation. Once the user chooses the desired ones, the IdP generates an assertion for TAAS containing the name values of these attributes (and only the name values, not the attributes themselves, hence ensuring privacy of the user).

When TAAS redirects the browser to the chosen IdP for authentication, it also requests the IdP for a randomly generated Persistent Identifier (PId) for the user, which is stored by TAAS and used as pairwise secret between TAAS and the IdP in order to identify the user in all subsequent communications between the two. If a new PId is received by TAAS at login time, a new entry for the user is created in its internal database. If a known PId is received, the locally stored entry from the database is retrieved and it is updated to reflect the current set of attributes. This also enforces user privacy, as this is the only identifier TAAS uses to identify the user, and since it was randomly generated by the authenticating IdP, TAAS doesn't store any information that may link the identity of the user to the user itself. In brief, TAAS only stores a table with the user's PId, his different IdPs and attribute value names attainable at each one.

Levels of Assurance

The IdPs authenticates the user with different strength levels. These levels are termed as Level of Assurance (LoA), level 1 being the weakest and 4 the strongest. Because the user can authenticate to

any IdP that has a trust relationship with TAAS, a *registration LoA* is stored when the user registers to an IdP. After the user authenticates to an IdP, the respective LoA is used as the current session LoA. To prevent escalation of LoA levels, other IdPs will only release attributes if their *registration LoA* is lower or equal to the current session LoA.

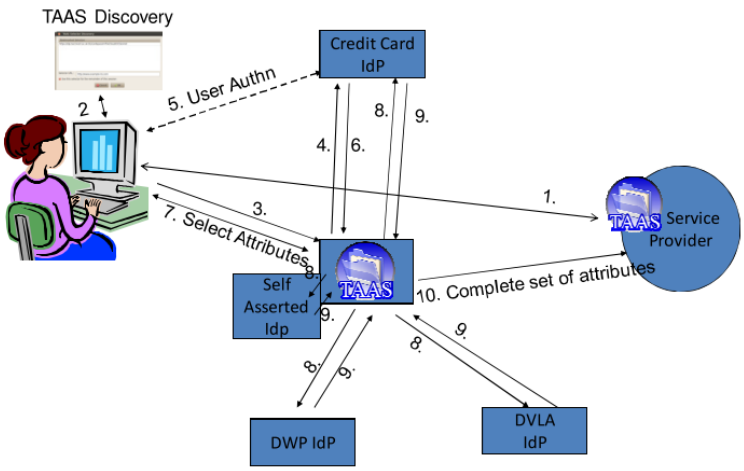


Figure 2.2: TAAS Protocol Description [2].

Operation Example

When the user wants to access a SP which requests any of his attributes, he is redirected to the TAAS login page where the requested attributes are clearly stated along with the security policy of the SP. After the user confirms, his TAAS discovery service plugin is activated (a thin client on the user browser). The plugin allows the user to select his preferred TAAS and is then responsible to secure the connections to it using TLS. If a valid user cookie session is already present in the browser, the authentication step may be skipped, otherwise the user is redirected to a WAYF service to display the IdPs this TAAS has trusted relationships with. The user chooses one of these IdP and is then redirected to it for authentication. After the user authenticates, the IdP generates an *authentication assertion*, which contains a random identifier used as a SSO token valid for any other entity that trusts this IdP, and a "referral" assertion containing the pre-linked PId that points to the user account in the TAAS service. TAAS can now show the user the cards (similarly to CardSpace (see section 2.3.2)) that hold the necessary attributes requested by the SP. The user then needs to choose all the required cards (some may be optional, depending on the SP policies). If no previously linked IdP matches the SP policies the user is prompted to do so by linking the necessary IdP to TAAS. After all the necessary attributes are linked, the user can finally select them.

Once the user has finished the selection and has clicked the send button, TAAS will query all the chosen IdPs for the users attributes. The user authentication assertion is also sent, so that the IdPs can verify if it comes from an authenticating IdP they trust. If the releasing IdPs trusts the *authentication assertion*, they cipher the requested user credentials with the requesting SP public key, digitally signs the message containing all the assertions and sends them back to TAAS for aggregation. These assertions must also all contain the same user PId. After all attributes are collected, TAAS sends them to the SP, that then proceeds to verify the *authentication token*, the IdPs signatures and retrieves the attributes using its private key. Figure 2.2 displays the TAAS protocol flow.

TAAS is an aggregation solution that uses the SAML standard, facilitating interoperability with other systems. Privacy of the user and security concerns have been addressed using transient identifiers, and by not storing the actual values of the credentials, but only meta-data.

A problem with the proposed solution is that at the IdP linking step, a user may provide the credentials

of other users (either maliciously or with consent), and aggregate attributes that aren't his. A solution for this would be to check the content of the attributes to find matches between different IdP's and give a certain assurance level if matches are found [32].

2.3.4 .NET Passport

.NET Passport is Microsoft suite of authentication services (with SSO support) for a number of devices and applications [33,34]. To access a service that allows authentication through the .NET Passport platform, the user must create an account through the suite's website, where he can then fill a number of attributes which are stored in a central database. The service claims to give the user full control over the disclosure of his stored information to the participating sites.

.NET Passport focuses on authentication only and lacks authorization features, hence, the SPs using the service need to store their own authorization rules. Besides this, if a SP requests information of the user that is not available through his .NET Passport account, this information will have to be stored by the SP itself. The system works through an authorized list of services (URLs) that may use the .NET Passport platform. These participating services must share a cryptographic secret with the platform in order to prevent unauthorized services from using the authentication functionalities.

When the user wants to access a participating service, he is transparently redirected to a .NET Passport page that might be co-branded with the SP. The user credentials are then sent securely to the .NET Passport server using an HTTP POST on top of the SSL/TLS protocol. The service also allows a stronger credential sign-in, which uses an additional PIN, with at least 6 digits, chosen by the user after trying to log in for the first time in a participating service that requests stronger authentication. Similarly to Shibboleth and CAS (see sec. 3.4.1) a ciphered cookie is used for SSO, hence removing the burden of the user having to login multiple times.

Microsoft .NET Passport was not very successful due to several flaws in the system design. On the security standpoint, it had a vulnerability that would allow an attacker to successfully perform Replay Attacks. Another significant problem is that the central database would be a single point of attack and failure and would give Microsoft control over all of the users data, which could be used without consent (e.g.: for targeted advertising). Users would have to put a lot of trust in Microsoft for disclosing personal credentials and sensitive information to its database. Because of these problems, the platform was considered a failure and was later discontinued.

2.3.5 OpenID

OpenID is a distributed open standard for SSO authentication based on URIs [35]. The user can obtain an OpenID URI by registering to an IdP that he trusts which supports the OpenID standard. The system is based on HTTP redirects, similarly to CAS (see sec. 3.4.1).

When accessing a SP that requires authentication, the user is requested to provide his OpenID URI. After the user provides the URI and confirms, the browser is redirected to the IdP chosen by the user for authentication, where he will insert his credentials and authenticate. The IdP then checks the credentials, and if correct, authenticates the user. He is then redirected back to the SP with a security token that asserts the user is authenticated, without sharing any of his credentials with the SP. Upon checking the assertion validity, the SP grants access to the user.

Even though OpenID provides an easy way to identify the user IdP for authentication it has several shortcomings. For OpenID to work, the SPs need to trust all IdPs that support it, and this might not be viable for all institutions. OpenID has also shown to have several security vulnerabilities [36].

2.3.6 Circles of Trust in Federated Environments

The concept of circles of trust involves the establishment of trusted relationships between partners that can lead to the federation of all user accounts [37, 38]. This implies the definition of well-defined frontiers delimiting a finite area for the federation and the identities that are part of it. This concept is very important for federated identity management because it involves multiple organizations, thus, an established federation among them must be well defined, comprehended and managed by the involved organizations, otherwise the security of the users and their identities and of the organizations themselves could potentially be compromised.

The current solutions of SSO across multiple services are usually restricted to within the same institution. As an example, Google allows SSO onto all of its services (Gmail, Google+, YouTube, Drive, etc. . .) once a user authenticates to one of them. A "Circle of trust" exists within the different services because they all belong to and are managed by Google. The creation of circles of trust to create federated SSO is much more difficult, not only because allowing SSO on a federation wide level requires that trust exists between all the involved organizations, but also because of interoperability limitations. Different organizations may use diverse authentication systems with different strength requirements. Stronger authentication is usually required when the threat of identity theft occurring has a higher impact on the SP. For example, if some user suffers from identity theft and the thief does transactions on the users behalf through his eBanking portal, besides the obvious consequences for the user, this might have catastrophic consequences on the client's trust in the bank's eBanking system. Therefore, banking or other sensitive transactions usually require stronger authentication means. This leads to different levels of trust and assurance between different organizations. These trust levels and federation level policies must be standardized.

Currently, trust relations are managed using the well known PKI infrastructure and its hierarchical structure of Certification Authorities (CAs) that issue the Public Key certificates and in which a high level of trust is deposited by organizations and users. This trust is also extended to all entities that present certificates issued by these authorities. However, this model is not suitable for cross-federation identity management, because some institutions might not necessarily trust all organizations that have certificates issued by some CA. Organizations might also issue their own certificates or have their own private CA. Individually managing each link of trust between organizations by exchanging public key certificates between all the trusted parties within a trust circle doesn't scale well. Because of this, three models for federation trust circles management have been proposed: The Hierarchical PKI model, the Cross Certification PKI and the Bridged PKI.

The Hierarchical PKI is the most simple and it follows the normal hierarchical structure of the PKI trust circles. A federation wide Certification Authority would be at the top of the hierarchy as the root CA and it would establish the relations of trust between the different, lower level CAs of each organization. This structure is inadequate though, because it would create a hierarchical structure between the circles of trust, meaning that the organizations either trust all certificates issued by the CA or none.

The Cross Certification model consists in establishing a cross-certification between all the different CAs that issue the certificates for their respective organizations. However, this would be hardly maintainable as the number of cross-certifications would grow geometrically with the number of members. With this model, certificate revocation would also be very difficult to maintain for the same reason.

The Bridged PKI is the proposed model to solve the problems that arise from the others. It consists in a bridge CA that creates cross-certifications with the CAs of each circle. The bridge CA is not a root CA, but its reputation is instead established by the number and respective reputation of the partners of the federation that establish trust links with it. This is different from the normal hierarchical model because instead of having to trust the root CA, the bridge CA's trust is based on confidence relations

between the partners. It scales better because each one of the federation members would only have to hold the bridge certificate (i.e: a single one instead of a myriad of them) with all the cross-certifications between the organizations. Moreover, this model allows to easily reject the trust with a partner, by revoking the certificate of the CA of that partner.

2.3.7 The Liberty Alliance

The Liberty Alliance is a standard organization, founded in 2001 by Sun Microsystems, whose goal is to enable secure online transactions among consumers, citizens, businesses and governments through the usage of open standards, while protecting the users privacy and security of identity information. The organization has participated in the development of standards such as the SAML 2.0 specification to facilitate federated authentication and attribute exchange, and also provides guidelines and best practices for IdM system management to maximize the privacy of the users and protection of their identity data. The Liberty Alliance is now called the Kantara Initiative[2].

2.3.8 SWIFT Architecture

The SWIFT (Secure Widespread Identities for Federated Telecommunications) project [3, 4, 39] aims at creating an IdM solution that overcomes the common shortcomings of the already existing ones. The project is based on a cross-layer approach that integrates network and applications layer from an IdM perspective. Even though some of the already discussed IdM solutions support federated authentication through the usage of SAML 2.0, most of them have shortcomings in the provided privacy or policy control. For the design of SWIFT, the following requirements were established:

Req. 1 - Overcome Identity Fragmentation: As already discussed, users identity is fragmented across several IdPs and SPs. Users need a way of having a unified view of their sparse identities across different information systems and providers;

Req. 2 - Cross-Layer IdM: SSO for network layer and not only application layer. The network layer is often neglected in current solutions, and if not taken into account, it can lead to several security and privacy vulnerabilities;

Req. 3 - Improved privacy Features: The identifiers used in the Network Layer can be used for correlation, hence why Req. 2 is relevant;

Req. 4 - Support for multiple devices: Multiple authentication tokens and devices must be supported due to the highly fragmented solutions already deployed;

Req. 5 - No dependency on online components: Guaranteeing 100% availability of IdPs isn't realistic and can render services unusable if they aren't reachable. It is needed to minimize/abolish dependency on IdPs availability.

Req. 6 - Backward compatibility: Provide compatibility for existing solutions and legacy systems, as it is not reasonable to assume that legacy systems can be replaced.

To achieve the implementation of the above requirements, the system architecture was split into the following components:

- **End User**: The entity itself - The user who wants to access services by making use of the IdM infrastructure;

[2] http://kantarainitiative.org

- **Service Provider (SP)**: The entity that will consume the user Identity information, if the user is properly authenticated and has authorization to the provided service;
- **Identity Provider (IdP)**: Provides the user required information to the SP, if the user allows him to. The IdP role is subdivided into three additional roles:
  - **Attribute Provider (AttP)**: The user identity information management functionality;
  - **Authentication Provider (AuthNP)**: Responsible for the user authentication only;
  - **Identity Aggregator (IdAgg)**: Manages the user Virtual Identities, which may aggregate attributes from several different IdPs. This component is important for Req 1.

The system relies on the definition of Virtual Identities (VID). A VID is a user's defined profile that aggregates authentication credentials and attributes from different IdPs. The VID is responsible for referencing an *authentication ID* (AuthNID), an identifier by which an authentication provider recognizes a particular authentication subscription, and one or more *Attribute IDs* (Attr IDs) held by different AttPs. Each VID includes information about the identity aggregator - the authentication providers, attribute providers, and identity domain where the VID was defined - and data to select the right virtual identity within that domain. To maintain user privacy, the system replaces the VID pointer with an encrypted artefact, and will only disclose the Aggregator ID to SPs.

Figure 2.3 shows the relationship among the framework elements. As showed in Figure 2.3, the different elements of the system only share *Pseudonyms* with each other and the SP only contacts the IdAgg, so that they cannot correlate a user's different VIDs to his real identity, hence keeping his privacy safe (as specified in Req. 3). Furthermore, a user is able to specify which attributes to disclose to specific SPs.

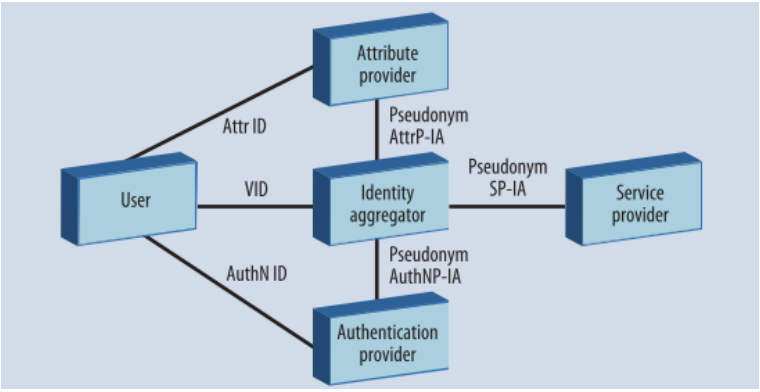


Figure 2.3: Virtual Identity Content [3].

Upon trying to access a protected service, the user selects one of his VIDs and a *SWIFT Initiation Statement* is created. This statement contains an artefact (that is meaningful only to the IdAgg) and the aggregator ID. The SP then uses a name resolution service (e.g. DNS) to identify the IdAgg, and redirects the user to it. The IdAgg then reads the statement and determines the AuthNP through the VID associated to the received artefact. The user is then redirected to the AuthNP, where will proceed to authenticate himself providing the needed credentials. Upon correct authentication, the user is redirected back to the IdAggs with an *authentication statement for aggregator* which also contains the AuthNP-IA pseudonym so that the IdAgg can associate the VID to the authentication without knowing the user credentials. Once the IdAgg does the association, a *SWIFT SSO* token is created and sent to the user. The SSO token also contains an *authentication statement for SP*. The user is then redirected to the SP with the SSO token, and the SP uses the statement to verify the authentication with the AuthNP. This

statement also contains the SP-IA pseudonym. The SP may then create a private cookie to allow the user to access its service without re-authenticating. If the SP needs to require additional attributes of the user, it may use the SP-IA pseudonym to request them from the IdAgg.

To improve on Req. 2, 3 and 4, some additional components were designed [4]. The proposed improvements are shown in Figure 2.4.

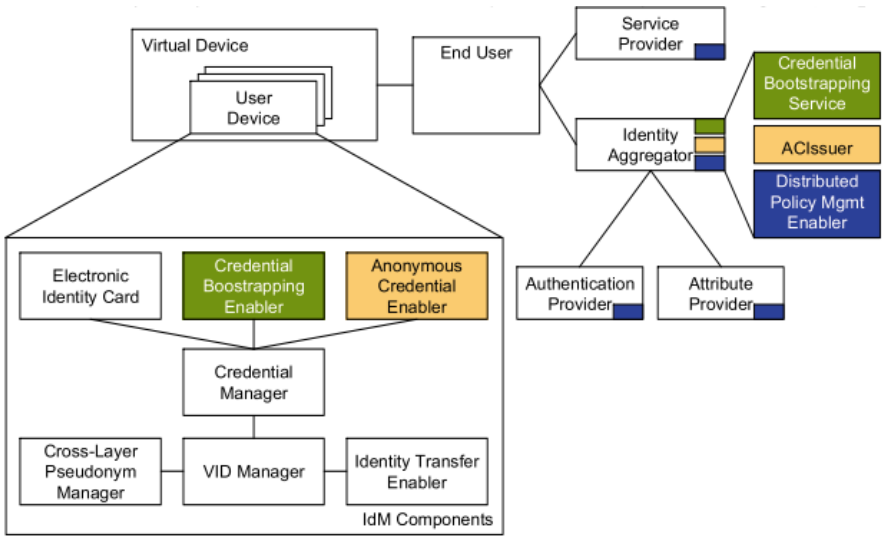


Figure 2.4: Additional components of the SWIFT architecture [4].

The VID Manager (VIDM) is a GUI that runs on the user device. It allows the user to manage his virtual identities and all IdM related tasks. The Credential Manager (CM) establishes connections with IdAggs and SPs and provides the user fine grained control over the IdPs credential release policies to the latter (Req. 3). The CM is also responsible for providing an abstract interface to Electronic ID Cards (EIDC) and the two other components of the solution: the Credential Bootstrapping Enabler (CBE) and the Anonymous Credential Enabler (ACE).

Using an EIDC gives the user more security, as it permits him to consume services from SPs without contacting IdAggs or IdMs for authentication, as he can use the card built-in cryptographic functions to authenticate directly to the SP by proof of possession of the EIDC and establish a secure channel with the SP.

The CBE provides backward compatibility by creating specific credentials to interoperate with other IdM systems. (Req. 6). The ACE gives the user more privacy by enabling the usage of anonymous credentials (using non-iterative zero-knowledge proofs[3]). This is valuable because using X.509 Certificates might provide SPs with more information than needed about the user, and upon multiple usages, they can be correlated.

The network stack employs its own identifiers that can be used to link the users pseudonyms between the IdAggs SPs. The Cross-Layer Pseudonyms Manager (CLPM) purpose is to solve this security flaw by using Virtual Network Stacks [40], which use different identifiers on a per identity basis (Req.3).

The Identity Transfer Enabler (ITE) function is to give the user with a unified view of all his user devices. This allows him to control his VIDS across his several devices in a consistent manner (Req. 4).

[3]zero-knowledge proofs are a method by which one party (the prover) can prove to another party (the verifier) that a given statement is true, without conveying any additional information apart from the fact that the statement is indeed true.

2.3.9 Project GUIDE

Project GUIDE is a proposed architecture for federated IdMs that establishes how circles of trust between entities should be organized [41]. The project required the affiliation of all participants in federations based on operational and formal agreements that would define the trust relations between them, in order to engage in transactions with their service users. At first, MSs started creating large circles of trust at a national level, but isolated at European level. Project's GUIDE objective was to create an architecture that would connect these isolated "islands", enabling these federations to unite in a major circle of trust, creating a single identity environment throughout the European Union. To achieve this goal, the system proposed the GUIDE Gateways. These gateways would sit on top of the existing national federations of the MS, establishing trust connections between them at a European level by supporting the different IdM and authentication technologies already in place.

2.3.10 Project MODINIS

Around the same time as the GUIDE project appeared, another pan-European model, called project MODINIS, was proposed [41]. MODINIS is only a conceptual framework where authentication levels and mechanisms across MS would be recognized and accepted between each other in accordance to a set of pre-established common criteria. The MODINIS conceptual Framework presented a viable European IdM system that was not bound to a particular architecture or to a specific technology, thus providing the needed flexibility for pan-European interoperability. However, it being a conceptual Framework, none of the implementation technical problems, such as semantic interoperability and reliability of the identity data, or which technologies to be used, were addressed.

It would be impossible to create trusted circle between all the IdPs of all the different countries, so the model relies in creating a national hierarchy of eGovernment Portals, where the one higher in the hierarchy in each country is to be established as the national wide portal, which would be responsible for communicating with the other national portals of other MS. This network of trusted national IdPs would be accessed through a Pan-European eGovernment Portal which citizens would use to access an IdP from another MS, which in turn, would connect to the lower level IdPs to get a user attributes. This architecture allows to create a layer of abstraction so that it is possible to create a circle of trust between MS countries in a scalable and maintainable way by delegating assurance to these trusted national portals.

2.3.11 Project PRIME (PRivacy and Identity Management for Europe)

PRIME is an European research and development project partially funded by the EU whose vision is to create an infrastructure with high accountability while respecting user privacy in their electronic interactions [5, 42]. The project consists of more than 20 partners mainly from Europe, comprising universities, public companies, a consumer protection authority and standardization bodies. The project not only focuses on the technical aspects of the problem, but also the legal and social aspects. Creating a trustworthy IdM system that gives users full control of their identity data so that it is not unknowingly shared without their strict consent are the main goals of the system. To empower the users with such control, the system has been designed from the start with these goals:

Keep the user informed and give him control: clearly identify the SPs and state what data of the user is disclosed to them, so they can have a clear view of all the transactions;

User Privacy Policies: When a user provides personal data, he can express his policies over the data handling and negotiate them with the SP;

Use only the minimum required data from the User A SP must only require data that is necessary for the realization of a transaction, and no more;

User Authorization: SPs can provide users with authorization after authentication, while keeping their anonymity;

User Anonymity: The project doesn't impose complete anonymity of its users, however, for low-risk interactions, the user may keep anonymity without disclosing any personal information. In high-risk transactions, cryptographic credentials may be used to improve accountability, but the user identity can only be revealed in case of dispute or fraud;

User accountability: To achieve the accountability mentioned in the previous point, SPs may be requested with user identity information in case of fraud or cheating, but personal information may only be disclosed in such circumstances.

System Parties: The system is composed of the Users, SPs and Certification Authorities (CAs).
System Tools: To achieve the listed goals a number of tools and technologies are to be used.

To accomplish non-repudiation so that malicious users can be held accountable for identity theft, cheating or fraud, the system relies on the PKI and trusted CAs.

To secure the communications between parties the TLS protocol is used. Anonymous communications are more difficult to provide, because the user identity may be linked to his MAC address, and therefore, special network precautions must be taken.

To keep the users anonymous (or partially anonymous) *pseudonyms* that only have meaning to the SPs are used to identify a user. So that he user can prove ownership of a pseudonym, instead of randomly generated, they can be cryptographically derived.

A credential is basically personal information of a user (e.g. his birth date or full name) that has been certified. Users can prove ownership of credentials through cryptographic means, such as X.509 certificates that are issued by trusted CAs, which is important for accountability.

System Architecture: The system resources are identified by URIs. This way, a user would have an URI associated to him, which would be used to identify his resources without disclosing his identity. To enable interoperability between the system parties, the W3C Resource Description Framework (RDF) is used. The semantic of the RDF expressions are of the form (*subject, predicate, object*). The RDF *triples* semantic definitions however needs to be standardized across the system so that the statements are understood by all the parties.

The system is composed by a central database that holds the user certificates, declarations (uncertified data, generated by some party), the default user policies and logs of the interactions with other parties. To control user access to the database, an Access Control component and an Identity Control (IC) are used, and a GUI is provided for IdM and policy control. On the SPs side, a component called the Obligation Management (OM) is used to enforce the user data policies and privacy. An overview of the PRIME architecture can be seen in Figure 2.5.

The database on the user's side is responsible for holding his data and credentials, and on the SP's side it stores the data of customers. The user accesses SPs through a web-browser. The middleware on the SPs side is responsible to enforce the policies negotiated between the two parties and restricting access to the attributes the user is willing to share with other trusted entities, according to his policies. Moreover, the middleware establishes secure communications between the parties and manages trustworthiness of the communicating entities. Only if a SP is able to prove sufficient evidence of trustworthiness, and agreeing on the user established policies, it will be granted access to the requested user credentials. On the SP side, the user is granted access to the resources he wants to access based on

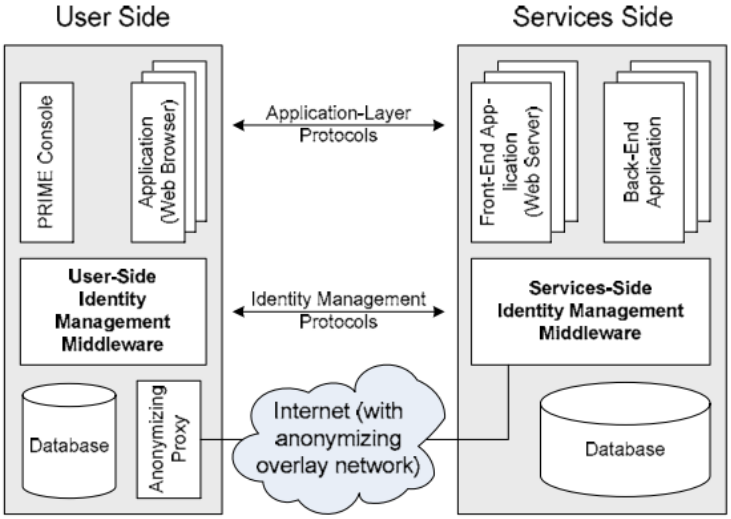


Figure 2.5: PRIME Architecture Overview. [5]

a attribute-based access control framework which computes access decisions based on the credentials he provides. For example, a user may only be granted access to a certain resource after he provides a credential that he is over 18 (and without the need of providing any other information). A flow of the interactions between a user and a SP may be seen in Figure 2.6.

The PRIME project's goal was to build a working prototype for the specified requirements described above, however, it never materialized. The research that was done within the project nonetheless provided useful developments that will be taken and used by other projects.

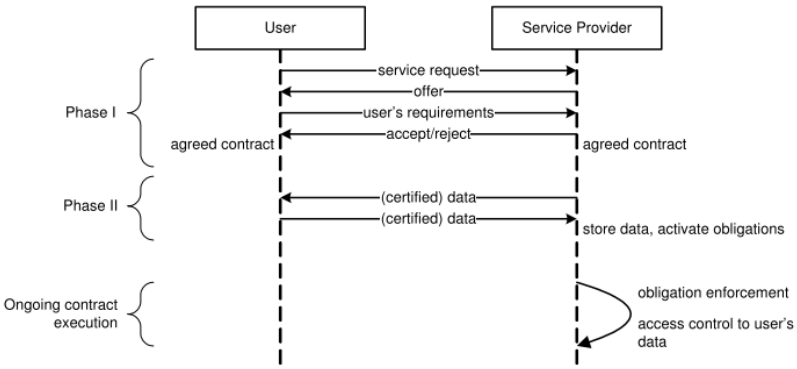


Figure 2.6: Execution of a transaction in PRIME [4].

2.3.12 Secure idenTity crOss boRders linKed (STORK)

The STORK project [6, 43] aims to solve the problems not addressed by MODINIS (see 2.3.10), while keeping the key features of the PRIME Project (see 2.3.11), allowing an European citizen to access eGovernment services and other SPs belonging to institutions and organizations residing in other MS, authenticating using an ECC card or some other token issued in his residing MS. This implies the roaming of a citizen's attributes across different European countries, creating a unified European electronic identification and authentication area.

2.3.12.A Architecture Components

The STORK architecture is comprised by two main components: the middleware and the PEPS (Pan-European Proxy Service). The former is a middleware that is installed on SPs and IdPs allowing them to communicate directly; the latter is a network of proxies that act as intermediaries for cross-border identity requests, creating links of trust between MSs and bridging the isolated "islands" that form the national IdM infrastructures. These two components can function seamlessly together, rendering the STORK infrastructure capable of handling both the centralized and distributed models (see 2.1.4). The data format used to exchange authentication and attribute information in the communications between PEPS and related entities (SPs and IdPs) is the Security Assertion Markup Language (SAML) standard (see section 2.2.4), and the messages are sent using a Representational State Transfer (RESTful) (see 2.2.3) approach, securing the communications over the TLS/SSL protocol

The PEPS Model

This model relies in a network of proxies established within the EU. At least one proxy per MS is necessary at a national level, which is responsible to bridge the national IdM infrastructure with the PEPS network of proxies. This allows establishing an interoperability layer on top of the different MSs deployed infrastructures that abstracts from the specificities of the individual solutions. This model scales well not only because of this abstraction, but also because it creates a system of trust relay, where each MS only needs to manage the trust relation between the national PEPS proxy and the existing national IdM infrastructure by inherently trusting in the network of PEPS proxies. Figure 2.7 shows an example of the trust circles that are created by the PEPS model and its basic process flows.

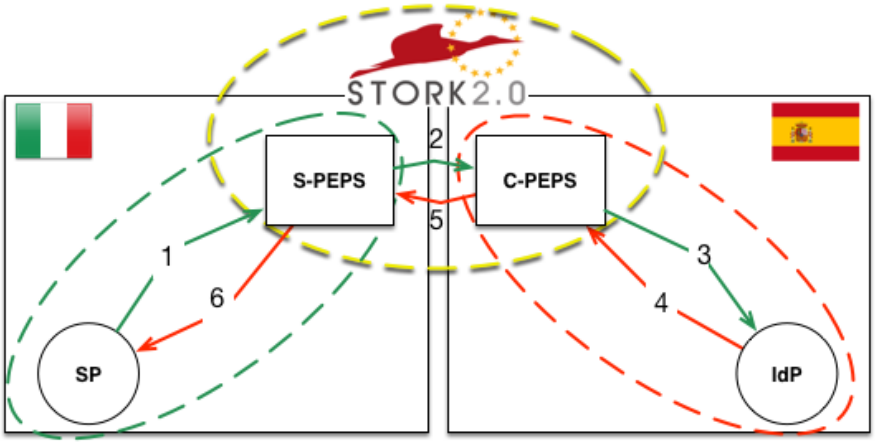


Figure 2.7: Trust networks in the PEPS model. In this example, an Italian SP is trying to access a Spanish IdP. The Italian SP trusts the respective national PEPS, that in turn trusts the Spanish PEPS, which trusts the Spanish IdP. This trust relay solution minimizes the management of trust relations, allowing the PEPS model to scale to a European level.

The process flow and basic function is also described in Figure 2.7. A PEPS server can either act as a so-called S-PEPS (PEPS in the state of the SP) or as a C-PEPS (PEPS in the state of the citizen). This nomenclature specifies the role of the PEPS in a given transaction: the S-PEPS receives an authentication request from a SP (1) and acts as the intermediary that routes it to the correct C-PEPS (2) (this is achieved by querying the user where he is from on the S- PEPS through a WAYF interface), that in turn is responsible to route the request to the correct national IdP (3). The IdP sends the response

back to the C-PEPS (4), which then sends it back to the requesting S-PEPS (5). The S-PEPS asserts to the SP that a user has been successfully authenticated on the IdP by sending back the SAML response (6), otherwise an error message is returned. If the SP also requested some user's attributes on step (1), the S-PEPS queries him if he authorizes such request before step (2), and when receives the response from the C-PEPS in step (5), prompts the user again to authorize the release of the attributes to the requesting SP.

The middleware Model

The middleware component enables the decentralized deployment of STORK. It has the advantage of providing better privacy and end-to-end security. Despite the advantages, the decentralized model faces scalability issues because it forces SPs to have to support the different foreign eID tokens that exist, which may be based on different protocols and technologies. To support these requirements, the middleware component has a modular and extensible design, so that the different IdM technologies support can be added through modules to the core component, called Modular Authentication Relay Service (MARS) or the (Virtual Identity Provider) VIDP core. The structure of the MARS can be seen in Figure 2.8.

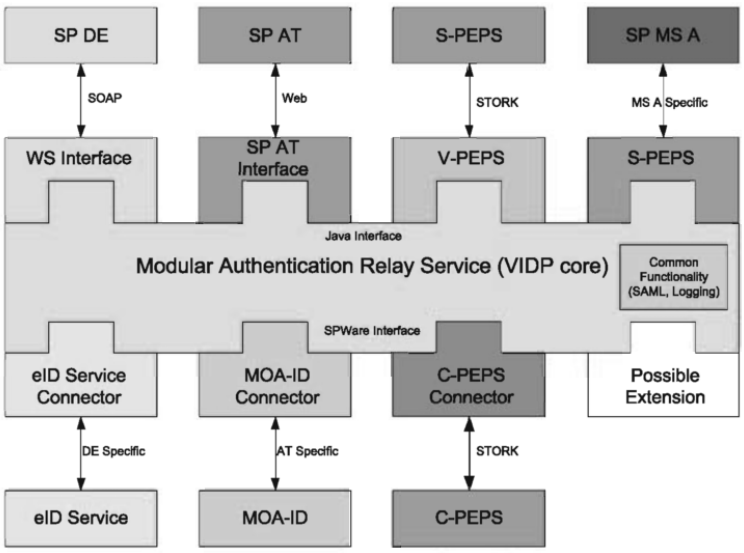


Figure 2.8: Modular Authentication Relay Service (MARS) [6].

The core component is responsible for handling the SAML messages, logging and runtime deployment of the extension plug-ins and plug-ons. These extensions allow the implementation of the country-specific national IdM components (also called SPWare) to integrate them into the STORK infrastructure. The plug-ons and plug-ins must implement two MARS interfaces, respectively: The Java and the SP-Ware interfaces. The former handles incoming authentication and attribute requests from the SPs by implementing the necessary protocols to interpret them (e.g: SOAP for the German SPs and REST for Austrian SPs), and route them to the latter, which is implemented by the SPWare connectors that are responsible for implementing the necessary technology to connect to the national-specific IdM infrastructure. Figure 2.8 illustrates the German and Austrian Java plug-ons, as well as the respective SPWare connectors for the German (eID Service) and Austrian MOA-ID IdMs.

As seen in Figure 2.9, the VIDP has a S-PEPS interface and a C-PEPS connector. This is because the middleware also supports the centralized PEPS model, and is in fact running in the proxies themselves, allowing interoperability between the centralized and distributed model – the mix model. For the

middleware to support both the distributed and centralized models, the following components are part of the middleware architecture:

V-PEPS: This plug-on is responsible for receiving STORK authentication requests messages from a S-PEPS.

C-PEPS Connector: The C-PEPS connector plug-in is the endpoint of the VIDP that routes a S-PEPS request to the citizen's country C-PEPS.

These components create the necessary conditions to enable full interoperability of the STORK infrastructure for countries that have either centralized or decentralized IdM architectures. As seen in Figure 2.9, Spain has a centralized model, against the decentralized approach of Austria. The diagram shows how the middleware behaves in 4 possible SP to IdP pair interactions:

Austria to Austria: In this case, the SP would contact the national infrastructure running the MARS middleware using the SP AT Interface. Since an Austrian citizen made the request, the VIDP would route the request to the MOA-ID connector.

Austria to Spain: Imagine a Spanish citizen wanting to access an Austrian SP. Once again the SP would make the request to the MARS middleware through the SP AT interface, but this time, because the IdP is in Spain (which uses a centralized infrastructure and relies on the PEPS model), the VIDP would route the request to the C-PEPS Connector, that would then contact the C-PEPS (acting as a stub) in Spain, that would then act as intermediary for the Spanish IdP using the SAML protocol.

Spain to Spain: For a national request within Spain, the national SP would contact the national S-PEPS to route the request to citizen's respective country (Spain in this case). The SP would create the SAML Request and send it to the S-PEPS running the VIDP, which would route it to the C-PEPS connector, redirecting it to the C-PEPS. Note that in this case the S-PEPS and the C-PEPS are essentially the same middleware running on the same machine, as the middleware would act as both entities in the case of a non cross-border use case.

Spain to Austria: Finally, in a Spain to Austria scenario, picture an Austrian citizen trying to access a Spanish SP would trigger the SAML Request to the national S-PEPS. Because Austria has a decentralized model, the Spanish S-PEPS would act a proxy/stub to the MOA-ID. Once the request reaches the VIDP through the V-PEPS interface, the MOA-ID connector would be used to directly connect to the Austrian IdP .

Note that in the above examples the route that the SAML Response would take has been omitted, as it would be the inverse of the request. The return flow has also been omitted from the diagram in Fig. 6 for simplification.

2.3.12.B Quality Authenticator Model

To create a single authentication and SSO area within the EU, mechanisms to control the quality and assurance of a given authentication need to be established. The variety of technologies used within the countries with functional IdM infrastructures leads to different quality levels of authentication and quality assurance depending on the technology used. For example, a username/password tuple is the weakest type of authentication, while cryptographic physical tokens (such as the various tokens with PKI features

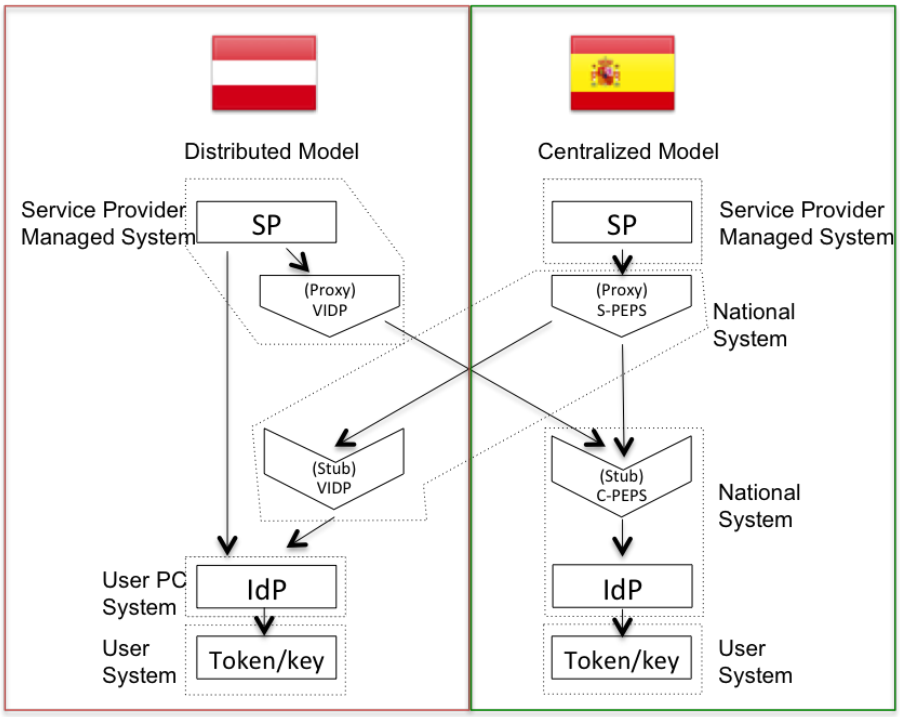


Figure 2.9: Mix Model – Distributed and centralized models integrated by deploying the stub/proxy of the distributed model in the centralized systems.

in Austria, or the ECC in Estonia/The Netherlands or Portugal), provide the strongest authentication method.

In such a diversified environment, it is important for SPs to be able to determine what is the strength of the technology that was used to authenticate a given user before authorizing access or consuming his attributes. This way, the SP can estimate the risk of giving access to a sensitive service depending on the strength of the authenticated technology used.

To solve this problem, the Quality of Authentication and Assurance (QAA) levels were designed as an important part of STORK. There are four QAA levels, 4 being the highest (authentication using an ECC) and 1 the lowest (authentication through a username/password tuple).

Because the attributes retrieved from the STORK infrastructure may be released from several different IdPs,they suffer the same assurance problems as the authentication methods themselves, as some IdPs may be more reliable than others, and the individual attributes may also have different assurance levels. For example, a university may release the address of a student, however, this information may be outdated, making it unreliable and lowering its assurance. If, however, this information was provided by a government IdP that retrieved the attribute from the national residents registry, it would have a higher level of assurance. The same may not apply to a student's diploma though, because this information is managed by the university itself, giving it maximum authority over its disclosure, which gives it the highest level of attribute assurance. To enable IdPs to specify the assurance level of the individual attributes, the Attribute Quality of Authentication and Assurance (AQAA) were designed, based on the QAA, and having the same four levels of assurance that directly map to the QAA levels. This permits that when an authentication is combined with multiple attributes, it is possible to make a separate statement of the AQAA level for all the individual attributes as well as the QAA of the eID level, giving SPs fine grained control over under what circumstances they would authorize a user or accept his attributes for a given operation, letting them have some control over the associated risk. In situation where a single AQAA level statement is to be applied to a set of attributes, the lower AQAA level within the set is to be applied.

This is because the weakest AQAA in the set defines its trustworthiness as a whole.

Let's consider, for instance, the following example: when a user tries to login into a SP that requires a set of attributes, it sends an attribute request that eventually reaches the C-PEPS. Once the user is redirected to his IdP and authenticates, the IdP will request the set of attributes to all the APs that might have them. The user's IdP will then respond to the C-PEPS with the received attribute sets from the possible various responses from the APs, each set containing, at least, the attribute names, AQAA, and values. The purpose of receiving the multiple sets (that may contain repeated attributes with the same values, but from different sources) is to find attribute matches in them so that the AQAA level can confirm and strengthen the user's identity by assuring their values from multiple trusted sources. This can also find possible mismatches, invalidating the attribute altogether by establishing that its value may be incorrect, fake or simply outdated. These matches can be searched through direct attribute name matches, or through Ontological Relations (ORs). After processing matches, the C-PEPS sends back to the SP the requested attributes and an AQAA of this aggregation process.

In situations where the user was not authenticated through the STORK infrastructure (e.g.: even though a user has a valid authentication statement, some AP won't use it and will ask the user to authenticate with his local credentials instead; or if a user authenticates to IdP A through STORK, but the attributes are obtained from IdP B, which trusts in A, through chaining), the obtained attributes AQAA and the QAA levels are to be removed. This absence indicates to the SP that no assurance level can be specified.

2.4 Conclusion

Despite the current evolution of IdM systems, technological fragmentation and interoperability are still complicated issues. There are also several legal implications that need to be addressed for attribute exchange and cross-federation authentication. As we have seen, several foundations and standards have arose from the necessity of interoperable and federated IdM systems. Another problem not yet completely solved is also attribute aggregation from different sources, as some level of assurance that the attributes being linked are actually belong to the same person. For federated IdM systems to be widely adopted, they need to provide a high level of assurance that its users privacy is of high priority, that their identity is secure from malicious attackers and that policies are enforced at the SPs, so that the users attributes and identity information is not misused or shared with other parties without consent and ensuring them that the risk of identity theft is low.

3

Background

Contents

This section reviews some concepts and technologies that are used in our solutions for the STORK academic pilot.

3.1 The Public Key Infrastructure

Most IdMs rely in the Public Key Infrastructure (PKI) [44] for authentication and authorization of the involved parties [15]. This is the case for the SAML [1] standard which is adopted by STORK.

The PKI relies on a pair of cryptographically generated keys, called the public and private keys (a.k.a asymmetric keys) and the use of public key certificates. Each key pair belongs to a specific entity and as the names suggest, the private key is private and cannot be disclosed to anyone, while the public key is to be shared with others. These keys are mathematically related so that data encrypted using one key can only be decrypted with the other. Moreover, it's computationally "easier" to generate keys and encrypt data than it is to try to find the decryption key given a ciphertext. The security of the keys depends on their size. Currently, the suggested size to be used for newly generated keys is of 2048 bits.

Let's proceed with an example that overviews how the PKI works: Imagine that Alice wants to send some data to Bob so that he can verify its integrity and authenticity. She will use a hash function to create a digest of the message, encrypt it with her private key, and send it alongside the message. Once Bob receives the message, he also calculates its digest using the same hash function, decrypts the received digest with Alice's public key and verifies if they match. If they do, it means that he can deduce that the message does come from Alice, and not some malicious entity, because only her could have used her private key to cipher the digest. Also, because the decrypted digest and the one calculated by Bob match, the integrity of the message has not been compromised. However, there's still a problem to solve: how can Bob trust that the public key he got from Alice is actually hers?

This is solved by Public Key Certificates: a Public Key Certificate is an electronic document that binds an Identity to a public key. This is done through a trusted party called the Certificate Authority (CA). The CA is responsible for creating public key certificates by asking the entity requesting the certificate to give some proof of her identity claims. The CA then creates the certificate using the X.509 format standard. X.509 certificates are basically data structures with a well defined set of data that binds the public key to the identity of its holder. After creating one, the CA calculates the certificate digest using a specified hash function and signs it with its own private key. By trusting the CA and validating some entity's public key certificate using the CAs public key (which is also distributed with a special certificate called the root), it's possible to verity the integrity and validity of that certificate, hence establishing that its contained public key indeed belongs to that entity, allowing to authenticate that entity or send it confidential information using the key.

The PKI also encompasses a number of other standards and protocols to manage certificate issuing, retrieval and revocation.

3.2 Node.js

3.2.1 JavaScript

JavaScript is an interpreted scripting language used to create dynamic web sites by running scripts on the browser that can interact with the content of the page. It has an event-driven nature thanks to its callback and anonymous functions model. This model allows the browser to queue function calls to the JavaScript runtime stack without having to block and wait for completion whenever one of such functions has to do lengthy I/O operations (e.g.: reading or writing a file to disk or waiting for user input). Instead, the functions are always tied to a callback (normally an anonymous function) that is pushed to a queue

(named the event-loop). Whenever the called function has completed (e.g: finished reading a file from disk). The event-loop queued callbacks are then pushed to the stack for execution whenever it is free. This model was chosen for the language mainly because of it was designed to allow an user to interact with the page and by using asynchronous callbacks, the browser can render the page without having to wait for an event to complete. JavaScript was designed from the ground up to be a client side scripting language and it remained the main use-case for JavaScript until recently.

3.2.2 The Framework

Node.js is a JavaScript framework that uses Google's V8 engine (the Chrome browser JavaScript runtime). It allows to build efficient and scalable event-driven web applications through asynchronous I/O calls by writing the server side code in JavaScript.

A node.js application operates on a single thread that uses the even-loop described above. The reality is that each I/O operation will run in a thread from an available thread-pool. However, this behaviour is hidden from the developer as he cannot explicitly interact with these threads and instead, he relies in the JavaScript callbacks provided by the language. This is different from other web applications frameworks that usually create a thread per client connection. This eliminates the overhead of creating different threads for handling multiple client connection, which is often the case in other web application solutions. The advantages of this are that:

- The programmer doesn't need to worry about multithreaded and concurrent programming, facilitating development and nullifying the difficult problems that come with it (i.e: deadlocks, starvation and inconsistency in data shared by threads).
- A Node.js server can normally handle many more connections than other multithreaded solutions.
- Less processing resources are used.

There are however some disadvantages to consider:

- Because the Node.js process runs as single threaded, the core business logic of an application is not usually executed in parallel, abating the potential use of multiple cores on a multicore processor.
- Creating CPU intensive business logic within a function is to be avoided. If a function takes a long time to execute within the stack, the event loop won't be able to push a callback to the stack until the function has returned, which can lead to performance problems.

3.2.3 Node Package Manager

The Node Package Manager (NPM) is an application dependency manager for JavaScript. Its use is not exclusive to Node.js but has been widely adopted by the Node.js community. The NPM command line tool is bundled and installed automatically with most Node.js framework installers. The packages that the NPM registry offers are hosted on GitHub repositories, meaning that the modules are open-source making it easy to contribute to a module's development, submit issues or just reviewing its code. NPM allows the developer to describe the dependencies of a JavaScript application in a JSON file called package.json in the root folder of the project. Using the command "npm install" in the root folder of the project will install all the necessary dependencies in a local folder called "node_modules", allowing their import into the application for usage in the code.

3.3 Simple Object Access Protocol

The Simple Object Access Protocol (SOAP) is a specification for the development of web services: "A software system designed to support interoperable machine-to-machine interaction over a network". The protocol was designed to allow the exchange of structured data in Extensible Markup Language (XML) format between different machines over the internet. The interface provided by a web service is defined in a file called WSDL. It is an XML structure that specifies a machine-readable description of how the service can be called, the expected data formats of the inputs and outputs of its remote functions and the location of the service's endpoint.

The specification supports multiple transport protocols, but the most widely used are HTTP and HTTPS. The reason for this is that using these transport protocols eliminates any potential problems with firewalls, as to allow access to a web HTTP/S server, requests directed to the well know ports 80 and 443, respectively, must be allowed through.

3.4 FenixEdu Academic (Fenix)

Fenix is an open-source[1] modular software platform for academic and administrative management of higher education institutions. It provides a back office system for schools to manage students, teachers, degrees and courses. It is written mainly in Java and is deployed in a Web Container such as Tomcat or Jetty.

The Fenixedu Academic deployment at IST relies on CAS for the authentication and authorization of users to the web application, so we will briefly describe it in the following subsection.

3.4.1 Central Authentication Service (CAS)

CAS is an open source Single Sign On (SSO) solution widely deployed across academic institutions, based on a RESTful architecture that uses HTTP redirects [45]. The system is composed by the CAS server and CAS clients. The CAS client is a Perl, Java, ASP or PL/SQL library which is easily deployed in any web application that requires authentication for authorization and access control.

When a user accesses an application that deploys CAS authentication, if he doesn't already have a cookie session in his browser, he is redirected to the institution's CAS server authentication web page. After the CAS server verifies that the application requesting authentication is in its allowed services list (stored locally by the server, and managed by its administrator), the user may introduce his authentication credentials. To ensure security, the CAS server must use HTTPS, so that the credentials are sent through an encrypted channel. After the user's credentials are verified (usually against some LDAP server or some other database), he is authenticated. The CAS server then generates a service ticket (ST) and automatically forms a new URL redirection that takes the user back to the application that requested the authentication. On the application side, the CAS client grabs the ST from the URL parameters, establishes a secure connection with the CAS server and waits for the server to validate the ticket. If the ticket is validated successfully, the user is granted access to the application and a session is established. The CAS server also gives the browser a private cookie that can only be shared with the CAS server that created it. This cookie prevents the user to have to authenticate again during its validity. This way, if the user accesses another application that uses the same CAS server for authentication, he is seamlessly given access. One of the key features of CAS is flexibility, since it supports a wide range of authentication mechanisms, from X.509 certificate, LDAP, Kerberos, OpenID and others. Moreover,

[1] https://github.com/FenixEdu/fenixedu-academic

CAS provides authentication proxying functionality. This is useful for applications or web portals that need to proxy the user credentials to access another external application to retrieve data.

CAS is a widely adopted solution, as it provides an easy implementations for secure SSO across an institution's services. However, the trusted services must be managed individually by an administrator, making the system focused on only allowing SSO within the institution circle of trust, i.e. the trusted services that are managed by the institution itself. The system strongest point is its wide supported libraries for the CAS client, that make it easy to deploy. A downside of CAS is that it is a single point of attack and failure.

3.5 The Portuguese IdM Architecture

The Portuguese national IdM relies on its infrastructure and the European Citizen Card (ECC) issued to all citizens as an authentication token. The eID SmartCard contains two digital public key certificates (see 3.1): one for user authentication and another for digitally signing documents. Both functionalities are protected by PIN codes that are sent to the citizens via a secondary channel (postal service) so that they are more difficult to compromise, as an identity thief would need to have access to both the eID and the PIN number to be able to commit fraud.

The national IdM and the Portuguese eID cards are managed by a governmental entity called AMA (Agência para a Modernização Administrativa). AMA developed a national IdM called SCAP (Sistema de Certificação de Atributos Profissionais - Certification System for Professional Attributes). The global architecture of SCAP can be seen on figure 3.1.

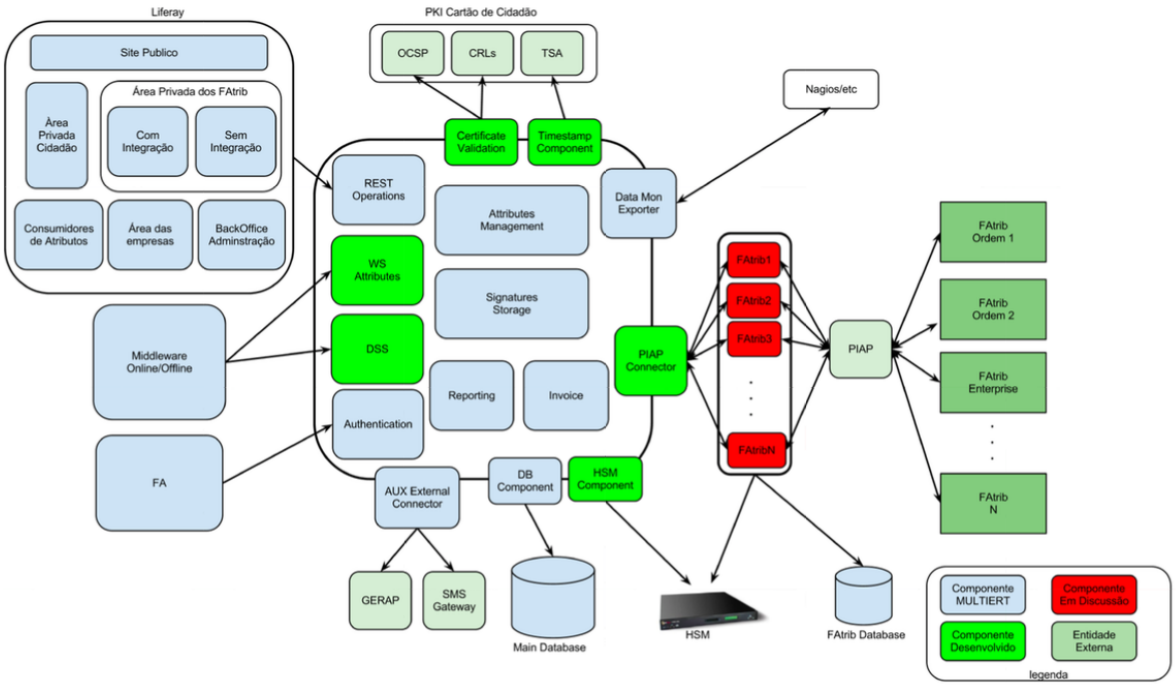


Figure 3.1: SCAP global architecture.

The goal of SCAP is to provide a national system to enable citizens to retrieve digitally certified attributes from other public institutions or APs. The attributes can then be used by service providers as credentials. An example would be the release of an attribute stating that a citizen is a civil engineer. This attribute is certified by another national institution called "Ordem dos Engenheiros" (OE) which is responsible for qualifying civil engineers with the entitlement to sign charters. With the SCAP infrastruc-

ture in place, a civil engineer could in theory sign a digital charter with his digital signature by providing the assertion that he is certified to do so by the OE.

As can be seen in Figure 3.1, the system is comprised by several different modules and frameworks, but on this document we will only discuss the most relevant:

- **FA:** Citizen authentication module;
- **PIAP:** Interface with APs module;
- **HSM:** The Attribute Signature hardware module.

The FA module allows a citizen to authenticate himself using his Portuguese ECC with a SmartCard reader plugged into his computer and having the provided middleware installed on the machine. When a SP requests authentication, the user's browser is redirected to the FA using the *SAML Web Browser SSO Profile* (see section 2.2.4). The FA web page loads a signed Java applet that is executed by the browser (after asking the user if he trusts the signature used to sign it and the applet itself). The applet then communicates with the SmartCard reader and prompts the user for his PIN number to get access to the ciphered information within the ECC.

A token with the user's identity is released from the ECC and sent securely to the FA through the middleware to authenticate the user. If some citizen's attributes were also requested by the SP, if they are available on the government's database (e.g.: the citizen's address or NIC) they are retrieved by SCAP, otherwise the request is relayed to the PIAP.

The PIAP is the component that establishes secure connections with all the APs using a SOAP web service. When an attribute request is received, the PIAP asks all the APs it knows for attributes, receives them and sends them back to Sistema de Certificação de Atributos Profissionais (SCAP).

The attributes aren't signed by the APs themselves, but instead, SCAP is responsible for the digital signature of the attributes retrieved from the possible several APs. The signature keys are established when an AP joins the infrastructure and are always stored by SCAP. This way the APs only need to worry about securing the connection with the PIAP, but not with private key management, which is all centralized and controlled by AMA.

This leads us to the function of the Hardware Security Module (HSM), an hardware component for cryptoprocessing that signs the attributes with the private key of the respective AP.

After all the above processes are terminated, the SAML Response is constructed with the signed assertions and it's returned to the requesting SP.

This overview doesn't include the integration with STORK, which requires, among other work, the deployment of the national PEPS and its connection with the infrastructure described above.

4

Solution

Contents

This section describes the proposed solutions for the different systems that compose the STORK academic pilot. For each one of the three systems we will describe the overall software architecture and its components in detail and proceed to an overview of how the systems fit into the national IdM and the STORK infrastructure. Finally we will briefly discuss the system.

4.1 Trusted Attribute Display Service

The Trusted Attribute Display Service (TADS) is a web application that relies on the STORK infrastructure (using the PEPS model) for authentication and attribute retrieval. The goal of TADS is to give European citizens a tool that allows them to materialize on paper the digital attributes that they retrieve from STORK, enabling the dematerialization of identity assured credentials with legal validity in a MS, and materialization into another, thus establishing a trust circle between digital and printed documents. TADS is to be deployed as a trusted SP, where users can retrieve and gather several attributes (primarily academic) from different APs and aggregate them into a signed, printable PDF document. They can then present a printed or digital version of the document to the verifier (whom is interested in verifying the document validity), which will access the TADS service for validation. If the verifier receives a digital version of the document, he only needs to check the validity of its digital signature. A Quick Response (QR) code on the printed documents allows the verifier to scan them using a webcam from the TADS verification service page and retrieve the digitally signed PDF document, thus allowing him to validate the document and the contained credentials. Figure 4.1 shows an overview of the involved entities and interactions in an exemplified usage flow.

An example of a document created with TADS is presented in Annex D.

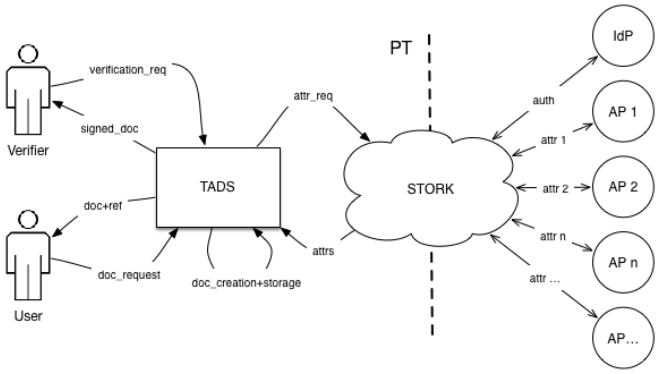


Figure 4.1: TADS interaction flow overview.

TADS is open-source and available on GitHub[1].

4.1.1 Software Architecture

The software architecture of TADS is presented in Figure 4.2. Node.js applications are composed by modules provided by the community that are managed through the NPM tool. The most relevant modules are represented in the figure. We will now proceed to describe those modules and other components in greater detail, as well as elaborate on the attributes that can be retrieved from the STORK infrastructure.

[1] https://github.com/STORK2/TADS

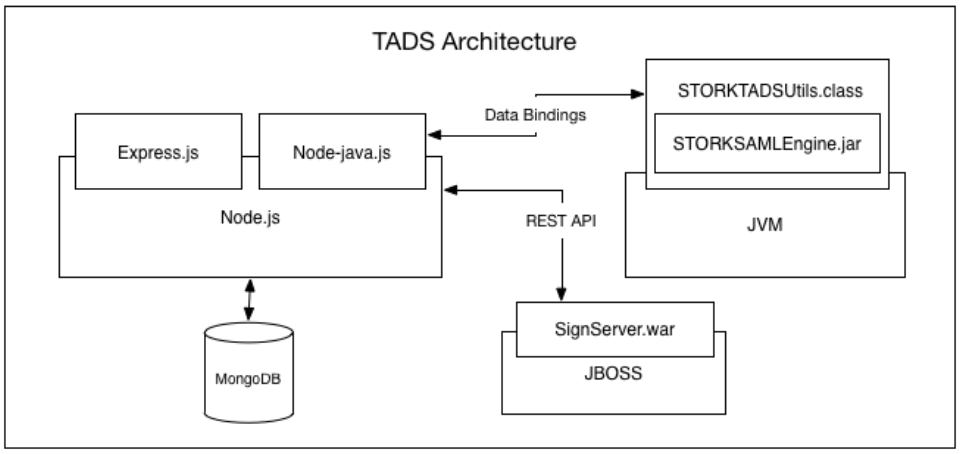


Figure 4.2: Architecture of TADS.

Simple and Complex STORK Attributes

TADS allows the user to create a PDF containing either simple or complex attributes retrieved from the infrastructure. Simple attributes are just key-value pairs, making it easy to interpret and insert into the PDF. Complex attributes such as the EDS however, are intricate XML structures that aren't in a human readable format. Because of this, they need to be interpreted so that its content can be arranged on the document in a way that the data is easy enough to be read by the users.

To uniquely identify the users in the system and keep track of their documents, the *eIdentifier* attribute is used, as it uniquely identifies a citizen within the infrastructure. For this reason, the attribute is always requested as mandatory alongside authentication.

Express.js

Express.js is a framework that provides Application Programming Interfaces (APIs) to develop web applications on top of Node.js following the Model-View-Controller (MVC) architectural pattern. It gives the developer the necessary tools to easily manage the web application routing and middleware components that handle things such as the views template engine, cookie and session management and serving static files.

MongoDB

MongoDB is a NoSQL document-oriented database that follows a schema-less model where documents are stored in structures that are very similar to JavaScript Object Notation (JSON). Like other NoSQL databases, it doesn't provide the classic ACID (Atomicity, Consistency, Isolation and Durability) guarantees of traditional transactional databases across different documents. A Document is the data model used in most NoSQL databases and is essentially a set of data that is stored within the database. Contrary to relational databases where the schema is well-defined at the time of the database creation, in this model the schema and the data has no clear separation. This means that the schemas are implicit to the JSON data structures being committed to the database, and they can be modified dynamically.

The upside of this model is that it allows more agility and makes the interaction with the database a lot easier for the developer. This is especially true for applications developed in Node.js because it uses JavaScript and its objects are represented in JSON, making it very easy to just reference some object and commit it to the database without ever worrying about schemas.

The downside is that because mongoDB doesn't guarantee ACID properties, multi-document transaction consistency is not guaranteed by the database. This is not a problem for TADS however, as multi-document transactions (i.e. transactions that involve multiple documents in a single commit) aren't necessary for the correct functioning of the application.

STORK SAML Engine Library

The SAMLEngine is a STORK Java library that provides an API for SPs to create and validate SAML Requests and Responses, respectively. The SP needs to configure the SAMLEngine in collaboration with the national entity responsible for the deployment of the PEPS, as it is necessary that the PEPS trusts the SP, meaning that the SP public key certificate used to sign the SAML Request needs to be added to the PEPS keystore, among other parameters regarding namespaces and the SP identifiers that are used to log the SAML transactions.

STORKTADSUtils class

As seen in Figure 4.2, the Node.js application doesn't invoke the STORKSAMLEngine library directly, but instead uses a wrapper class created specifically for TADS. This is to expose a simpler API to interact between Node.js and the library. Moreover, this wrapper ensures that the STORKSAMLEngine is instantiated only once upon deployment following the singleton design pattern. This way TADS always sends the SAML Requests and Responses to create and validate to the same STORKSAMLEngine instance, reducing overhead and improving performance.

Besides the wrapping API for the STORKSAMLEngine, the wrapper also provides function calls to access some Java bindings for the EDS to extract the values of the different fields that comprise its complex XML structure.

Node-java Module

Because the provided SAMLEngine library is in Java, this node-java module is necessary to be able to bridge the Node.js application and the library API. The module allows a Java class to be instantiated from the Node.js application and creates the necessary data bindings to directly call its methods and retrieve variables that are in a Java primitive data type to the Node.js environment, automatically converting them to the JavaScript equivalent. This allows to instantiate the Singleton instance of the STORKSAMLEngine in the Java Virtual Machine (JVM), and to invoke the wrapping API method to create the SAMLRequest, retrieving it on the Node.js application as a base64 string, which is then inserted into the HTTP POST message to be sent to the national PEPS server.

SignServer

SignServer[2] is an open-source application framework written in Java that allows to perform a wide range of cryptographic operations for other applications. In IST's deployment of TADS, it acts as a local server deployed on a JBOSS application server instance, exposing a REST API that is invoked to use its cryptographic features. The only SignServer feature that is used by TADS is the PDF signature function. When TADS creates a new PDF containing the user attributes retrieved via STORK, it sends it to SignServer to add a cryptographic signature to it before storing it to disk. This enables a verifier to check the document integrity and authenticity using any PDF reader application that supports digital signatures and displays this information to the user.

[2] www.signserver.org

The private key used by SignServer is the same used in the TLS certificate to establish HTTPS connections to the TADS server. This way a user can easily see that the PDF was issued by TADS itself.

4.1.2 Interaction flow

TADS gives the user the following three options to an user accessing it from his web browser:

- Create Document: To create a new signed document;
- Manage Documents: To manage documents the user has already created;
- Verify Document: For a verifier to access the QR code scanning utility to retrieve a signed PDF document.

A Figure of the TADS landing page where the user is presented with these options can be seen in Annex A Figure A.1.

The interaction flows for both a user and a verifier are explained below.

User Interaction

If the user selects the *Create Document* option, he will be presented with the attribute selection page. There, the user must select his country of origin and the attributes that he wants to request as: mandatory (meaning that if those attributes cannot be retrieved, the operation will fail); as optional (meaning that those attributes will be retrieved if available); and the ones not to request. After selection, the user clicks on the *Submit Request* button, the SAML Request is created and the user is redirected to the national S-PEPS with the SAML Request within the HTTP POST as a parameter, as well as the country code of his country of origin. Once the browser redirects to the S-PEPS, the user sees a page where he can confirm the attributes to be requested as optional or mandatory and proceed to confirm that the request can be sent to the C-PEPS of the user. The S-PEPS knows to which C-PEPS to redirect the request using the country code he receives in the HTTP POST request. Finally the browser is redirected to the C-PEPS whom will forward the browser to the national IdP, where the user must authenticate using his national credentials (in the case of Portugal, he will use his citizen card with a SmartCard reader attached to the computer). After successful authentication, the SAML Response is sent back to the C-PEPS with the attributes and the authentication token. The user can verify what attributes and respective values have been released by the APs and proceed to confirm the forwarding to the S-PEPS. The S-PEPS verifies the validity of the SAML Response tokens and sends it back to TADS. The SAML Response is validated, and if the authentication token is valid, the PDF document is created and signed. The user is then displayed the document management page, where he can see the list of documents he created, delete, or open them for disk storage or printing.

If the user wants to access his stored documents in a new browser session, he must click the *Manage Documents* button in the main page. In this case, no attribute selection page is displayed, but instead a SAML Request with only one mandatory attribute (*eIdentifier*) is automatically created and sent to the S-PEPS. Once the user authenticates and the SAML Response comes back (and its validity is checked) the user gets access to the document management page.

Verifier

When a user presents a printed documents created via TADS to, for example, a possible employer, he must be able to verify the validity of the document. To do so, the employer (the verifier), can go to the

TADS page and click the *Verify Document* button. Once he does, he must insert his e-mail info a form and submit. He will then receive an e-mail with a link that will give him access to the verification page (this e-mail step was added for logging purposes). On the verification page, the user will be requested to give the browser access to the computer webcam. A form also exists so that the verifier can insert a UUID to get the original signed document in case the webcam isn't used for some reason. If access to the webcam is granted, all the verifier must do is face the QR code on the printed document to the webcam, and if the document was indeed created by the TADS and is available in its database, it will be retrieved and displayed within the browser. The user can then store it or open it with some other PDF reader to check its signature.

4.1.3 Security

TADS relies on TLS/SSL connections using the HTTPS protocol to secure the connections with the clients (the browsers), this means that the Node.js server serves a public key certificate containing its public key and signed by a trusted CA. During the TLS handshake the certificate is presented to the client, which can be verified following the certificate signature chain, and if the client trusts the CA and the certificate is valid, the TLS protocol begins and the HTTPS connection is established, preventing Man-in-the-Middle attacks and protecting the exchanged information's privacy and integrity.

The SAML Requests tokens are signed using the same private key that was generated for the TLS public key certificate. This means that for the PEPS to accept the SAML requests that TADS sends it, its public key certificate has to be added to the S-PEPS trust keystore. This has to be done by the administrators of the S-PEPS server (AMA in this case). The same thing has to be done for the SAML Responses that the national PEPS sends back to TADS, meaning that the public key certificate of the S-PEPS has to be added to the SAML Engine keystore used to validate the SAML Responses. However, the SAML tokens are only signed, and not cyphered, this means that the attribute key and value pairs are in plain XML. Again, the SAML Requests and SAML Responses are exchanged through HTTPS to cypher the communication channel, ensuring the confidentiality of the messages.

4.2 eLearning Application

The eLearning pilot at IST is integrated with the institution's academic web platform (FenixEdu-Academic) and the CAS (see section 3.4.1) deployment. The system will allow a foreign student to access an online course on the Fenix platform by authenticating at the IdP of his country of origin and obtaining the necessary credentials that prove his enrolment in the course through the STORK infrastructure, granting him access to its contents and resources.

We will now describe the software architecture of the eLearning application.

4.2.1 Software Architecture

Figure 4.3 shows the overall architecture of the application.

The involved entities are the following:

- Fenix-edu Academic platform: This is where the course will be created and hosted. The platform provides the tools to maintain and manage all the necessary information to run the course and its contents;
- Central Authentication Service: This is the SSO server deployed at IST that is used for access control and authorization for all the services offered by the informatics department, including Fenix.

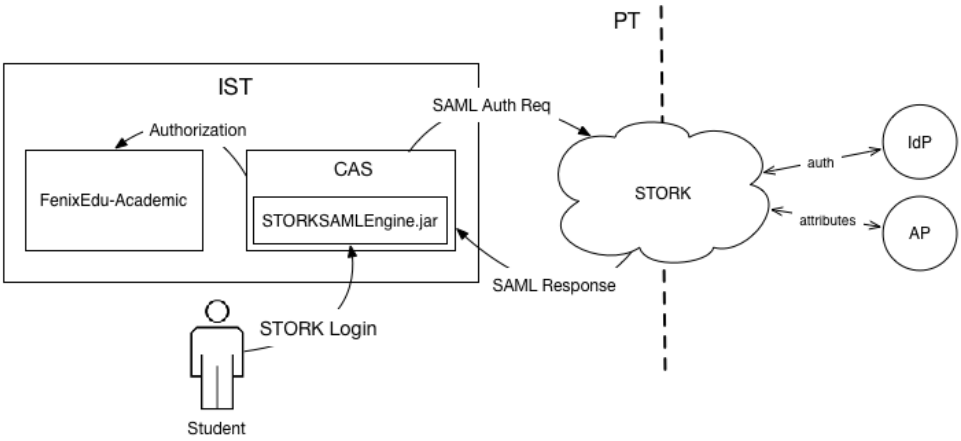


Figure 4.3: eLearning architecture overview.

This means that it has to be connected to the STORK infrastructure to make it accept STORK attributes as credentials and grant the user access to Fenix and its eLearning course;

- The national PEPS and the STORK infrastructure.

Academic is one of the modules of the FenixEdu software platform and it allows to manage curricular units within the university. The platform is accessible from a web browser by students and teachers giving them access to interfaces that allow them to see and manage the courses contents.

For the eLearning pilot, a course called STORK101 has been created on the platform. Even though Academic offers an extensive set of functionality for the management of courses, it doesn't provide an access control model based on capacities. When a new student registers at the university, he must personally go to the school administration services and present the necessary compulsory documents. He must then proceed to fill some forms on a computer, where part of the information is electronically gathered from the Portuguese ECC with a card reader. Among the registration steps the student account is also created, which will allow him to access the Fenix platform and other services. Once the registration forms are submitted, the student his given an ISTID (an unique identifier) and his identity information and login credentials are stored in the institutional Lightweight Directory Access Protocol (LDAP) server. This server is responsible to keep all the personnel's (including students, professors and everyone with an ISTID account) roles, access privileges, login credentials and identity information.

All access control is done via the CAS server, where users must authenticate to access services. Credentials are validated by querying the LDAP server for the entry with the corresponding ISTID, and checking if the credentials given for authentication match the ones stored in the server.

The flow described above however doesn't work for a STORK student trying to access the eLearning pilot, as he won't have any records in the LDAP server because he never went through the registration process of a regular student. Instead, he is a student of another institution accessing the course content presenting the necessary STORK attributes and identity credentials.

To allow Academic to grant access to an external eLearning student via STORK without changing the core business logic of the platform, two steps were required:

1. Develop the necessary changes for the CAS server to be able to create and validate SAML Requests and SAML Responses, respectively;

2. Develop the necessary auxiliary classes and an API on Fenix to allow the creation of a student on-the-fly, so that he can be identified by the system and logged into a session.

Central Authentication Service STORK Integration

The CAS server currently deployed is at version 4.0.0 and it uses the Spring MVC/Webflow framework. The vanilla version of CAS already offers a number of Authentication Handlers for easy integration with validation against LDAP servers and a number of other alternatives. However, it does not offer an out-of-the-box solution that works for the SAML assertions used within the STORK context. Because of this, it was necessary to develop two different web flows using the Spring framework:

The SAML Request flow: This is the web flow that allows a user to begin the authentication process via the STORK infrastructure. This implies the addition of an option on the CAS login page that users can select if they want to authenticate using their European eID credentials. Moreover, because the S-PEPS needs to know to which country to redirect the SAML Request, a WAYF page where the user selects his country of origin was added to the CAS user interface. Finally, a controller that constructs the SAML Request and sends it to the S-PEPS by redirecting the user's browser has also been created.

The SAML Response flow: Is the web flow that validates the SAML Response when the user is redirected back to the CAS server after going through the STORK authentication and attribute retrieval flow. The SAML Response comes within the HTTP request as an URL parameter. The necessary web flow was set up so that if a SAML Request parameter is detected, it is routed to the custom authentication handler created for the SAML validation.

In both flows the provided STORK SAMLEngine library is used to both create and validate the SAML messages.

To develop the SAML Request flow, it was necessary to add the option "e-ID European Credentials" to the drop-down menu, as seen in Annex A Figure A.2. To do so, the CAS login page was modified and an entry was added to the menu that redirects to the newly created view for the country selection (Annex A Figure A.3. When the user selects his country of origin and clicks the continue button, the STORK authentication process is triggered and the action that invokes the SAML Engine to create the SAML Request is called. From here, two different SAMLRequests might be created, depending on the country the user as chosen. The flowchart on Figure 4.4 shows the flow of the creation of a SAML Request.

The check of whether the country of origin is Portugal is done to create a SAML authentication request where the only mandatory requested attribute is *IdentificacaoCivil* (a national unique identifier called NIC). This attribute is not part of the STORK defined set, but it is supported by the Portuguese IdM and can be used to authenticate Portuguese IST students using their ECCinstead of the username and password tuple.

If the country of origin selected at this step is any other, then we can assume that it is a foreign student trying to access the eLearning pilot. In this case, the SAML Authentication request is created with a number of mandatory fields: *eIdentifier*, *givenName*, *surname*, *gender*, *eMail* and *textResidenceAddress*. These attributes are the minimum required set to create a new user in the Academic platform. The SAML Request is then sent to the national S-PEPS so that it can be forwarded to the user's C-PEPS.

The SAML Response flow is a bit more complex (see Figure 4.4). When the user is redirected to the CAS page with the SAML Response into the HTTP parameters, a condition into the Spring web flow configuration will check if a parameter called "SAMLResponse" exists. If so, it will invoke a class called "STORKNonInteractiveCredentialAction", which is responsible for extracting the SAML Response from the HTTP Request and validate it using the STORK SAMLEngine instance. If it is valid, a StorkCredential instance is created, containing the SAMLResponse itself and the SAML AuthnResponse that contains the user's authentication token and his retrieved attributes. Next, the CAS server searches for an Authentication Handler that supports the specific StorkCredential and will find our custom handler.

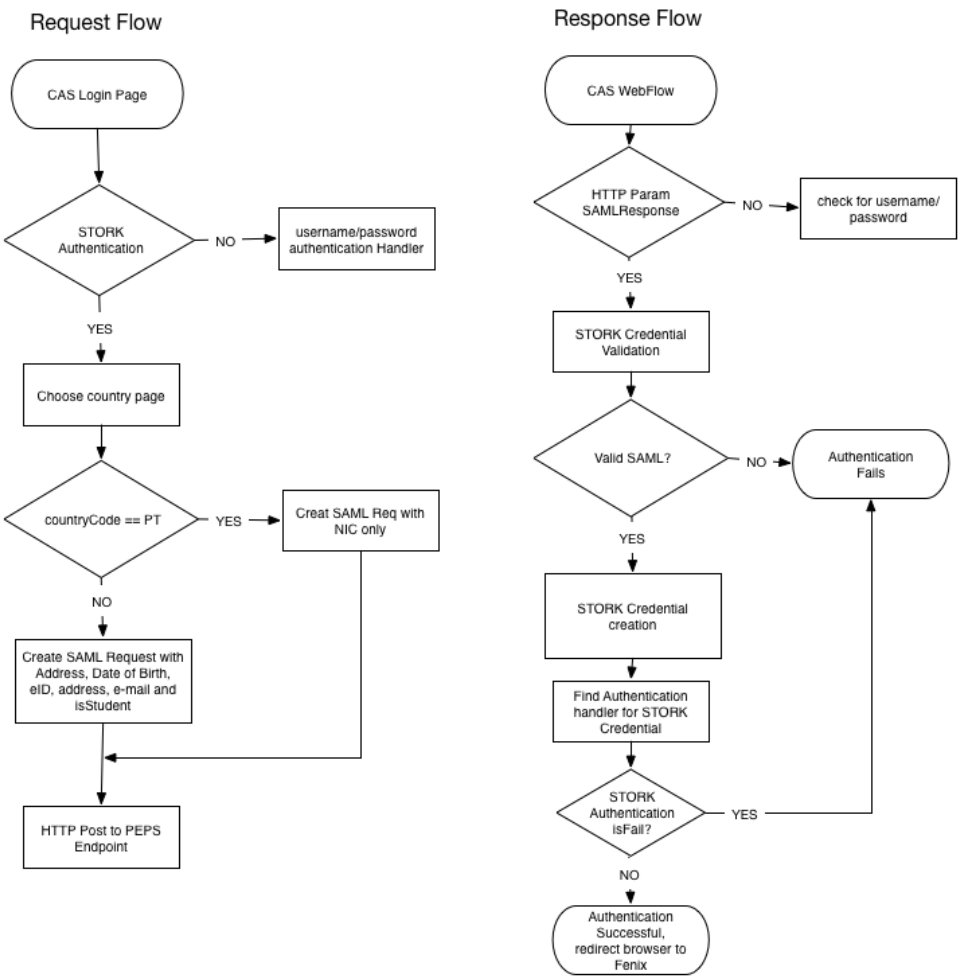


Figure 4.4: CAS SAML Request and Response flows.

The "SAMLResponseAuthenticationHandler" instance is responsible for verifying if the SAML Response authentication token is valid, by checking a variable that is set as true by the SAMLEngine in case the authentication at the IdP failed. If the token isn't valid, the CAS server throws an exception and doesn't authenticate the user. If it is valid, the handler checks if the response contains the *IdentificacaoCivil* attribute. If so, it means that the user is an IST student trying to authenticate with his citizen card. The handler calls a REST API method on Academic that searches for the ISTID with the corresponding NIC. If it doesn't exist, Academic returns a 404 code and the authentication fails, otherwise the IST student identifier is returned and the student is authenticated using it as the corresponding principal. However, the *IdentificacaoCivil* isn't found in the retrieved attributes, it means that the user is an eLearning pilot student trying to access the course. In this case, the handler checks if the *isStudent* value of the sub-attribute *course* asserts that the user is enrolled in the STORK101 course, and if so, calls another method from the REST API that: 1) checks if a user with the provided *eIdentifier* already exists, and returns (with HTTP code "200 OK" in the response) and its ISTID if it does; and 2) if it doesn't, a new student is created with the information provided by the mandatory requested attributes, and returns the newly attributed ISTID back to the CAS server (with HTTP code "201 CREATED" in the response), which will the proceed to authenticate the newly created user with the new ISTID as its principal.

FenixEdu-Academic Integration

Creating and enrolling the student in the STORK course on the Academic platform requires a number of steps. Figure 4.5 shows the UML for some of the most relevant classes involved in the process.

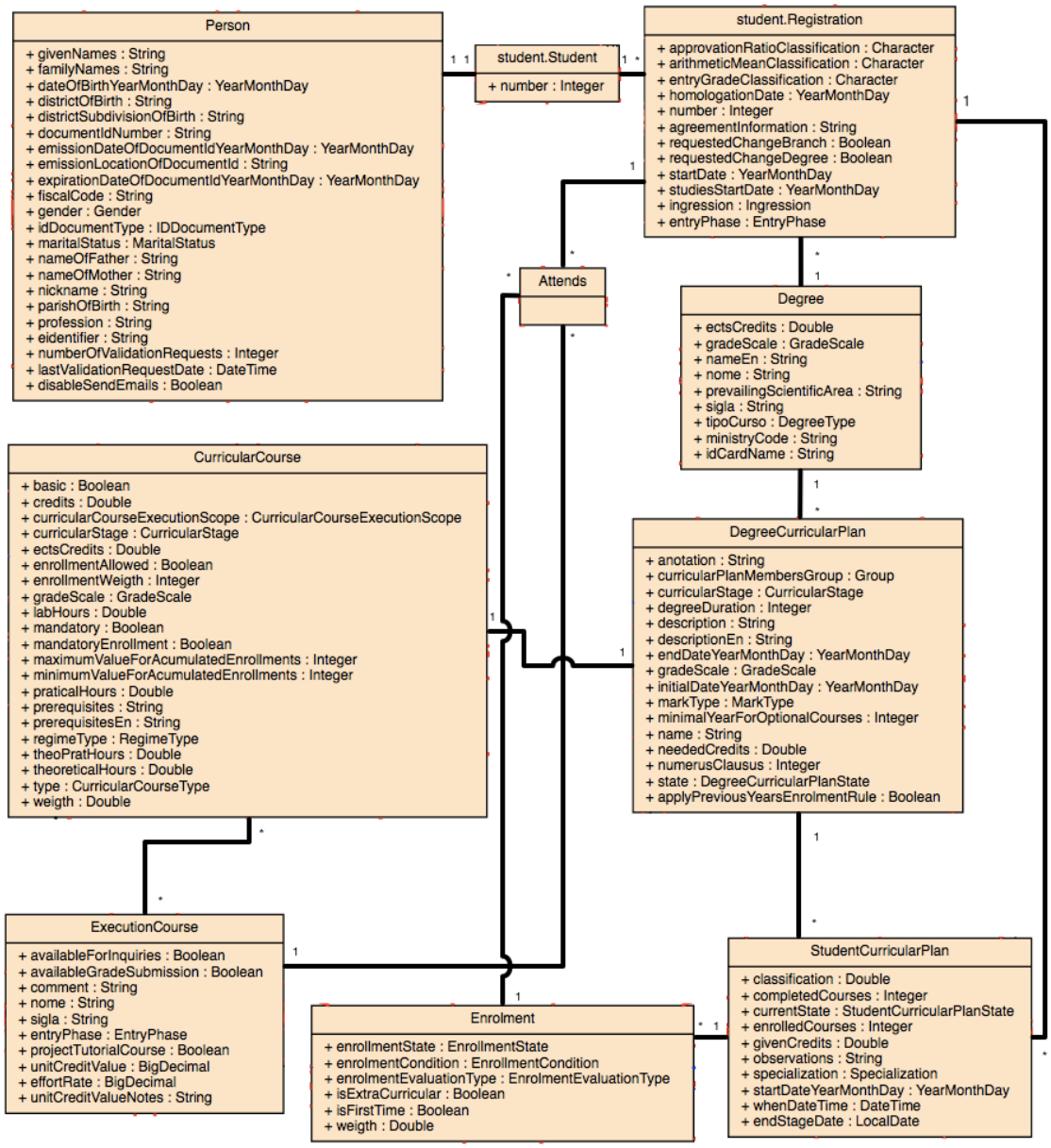


Figure 4.5: The UML of the data objects associated to a eLearning student creation and enrolment.

The first necessary step is to instantiate a new Person object passing some of the STORK retrieved attributes (gender, email, eIdentifier, date of birth and address) that describe the subject to its constructor. Once the Person object is created it needs to be linked to a Registration object. To do this, a new Registration has to be created for a given degree. The degree of the Registration has to be the one where the STORK101 course has been created and is available for enrolment. Finally, the new user has to be enrolled in the course. To do this, a reference to the STORK101 ExecutionCourse object has to be retrieved and a method is called that creates the "Attends" object and the necessary references automatically. Figure 4.5 shows the minimum amount of objects and relations that need to exist to enrol the student in the course. Some of the objects and references are automatically created by domain classes, these details have been omitted for simplicity, but it gives an idea of the complexity of the underlying

data model. No simpler way exists because the platform doesn't support capacities and on-demand enrolments. This makes sense because of the business logic related to the University business processes, so a workaround had to be found to allow the procedure to be synchronous, as it is a requirement of the pilot.

Authentication is of the sole responsibility of CAS, for this reason, the SAML Response validation has been delegated to it. However, for it to be able to create a new student "on demand", Academic has to expose an API to allow it, because all the institution's student information is stored and managed by it. The API follows a RESTful architecture and is secured by a username/password tuple for access control and by using HTTPS. Two distinct functions are provided by the API:

The student creation and enrolment function: The function is accessed at path "/fenix/api/fenix-ist /ldapSync/stork/createperson", and expects to receive a JSON object containing the following parameters: *eIdentifier*, *givenName*, *surname*, *gender*, *eMail* and *textResidenceAddress*. The function searches for a student entry that has an IdDocument whose value matches the received *eIdentifier*. If a match is found, the ISTID is returned as a string in the entity body HTTP field with code "200 OK", otherwise, the student is created using Academic domain methods providing the remaining attributes as parameters. Because no actual value is retrieved from STORK for the IdDocument object associated to the Person, the *eIdentifier* is stored as the IdDocument value of the Person object created for the new eLearning student so that it can be identified and retrieved. Finally, a new ISTID is returned in the HTTP entity body response with code "201 CREATED".

The NIC lookup function: Accessed at path "/fenix/api/fenix-ist/ldapSync/stork/checknic", it expects a JSON object containing the NIC of the student. A domain function is called to search and retrieve a Person object providing his NIC value. If a match is found, the corresponding ISTID is returned in the HTTP entity body with code "200 OK", or an empty string with HTTP code "404 NOT FOUND" otherwise.

Once the student has authenticated at the CAS server and is redirected to the Academic platform, a filter intercepts the HTTPS request and looks for the generated ticket in the parameters. Then, the CAS client establishes a secure connection with the server to validate the retrieved ticket and upon successful validation, the client receives the principle with the ISTID of the user. Now that Academic can identify the authenticated student, a browser SSO session is established, so that during the validity of the current session the user doesn't have to sign in again to access his personal area on the platform. In the case of a STORK student, he will be able to access the contents of eLearning course, while IST students can access the courses they are currently enrolled in.

4.2.2 Interaction Flow

Figure 4.6 shows the overall system architecture. We will now briefly describe the involved flows and interactions.

For the exemplified flow, lets assume that a student from Spain wants to access the eLearning pilot course at IST. In the 1st step, the student has followed some URL that sent him to the Academic web platform. There, the browser is automatically redirected to the CAS login page (step 2), where the student can select the "e-ID European Credentials" authentication method. Upon agreeing with the regulations and clicking continue, a WAYF interface is displayed where the user selects his country of origin (Spain, in this case). Once the country selection is submitted, the browser is redirected (step 3) to the national S-PEPS). The S-PEPS will then redirect the requested (step 4) to the Spanish C-PEPS. The C-PEPS shows the user the requested attributes, and asks him to authorize the request. Once

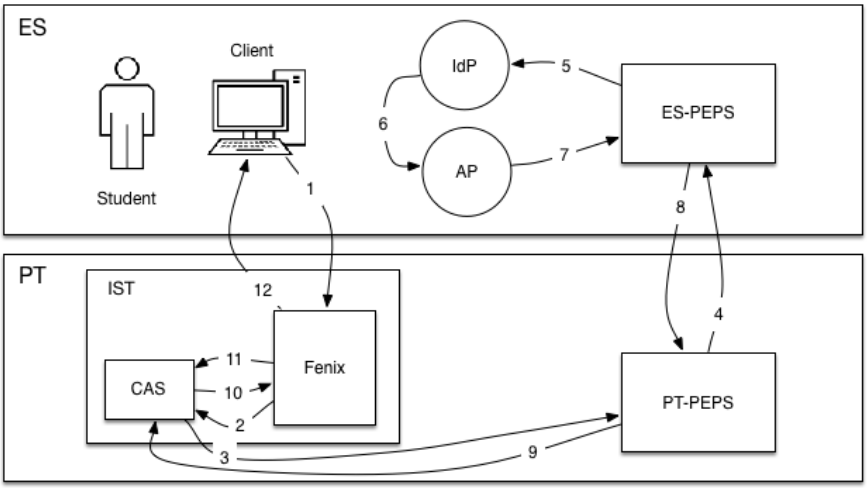


Figure 4.6: eLearning architecture interaction flow.

he confirms, he is redirected to his IdP (step 5), where he must authenticate, and if successful, the SAML Response with a valid authentication assertion is created. Now, the IdP can forward the SAML Response to an AP (step 6) that can add the requested attribute assertions of the user by validating if the authentication assertion is valid. Once all the assertions have been added to the SAML Response, the AP redirects the browser to the C-PEPS with it (step 7). The C-PEPS then inquires the student for the approval for the release of the retrieved attributes to the SP and redirects him back to the S-PEPS (step 8). The S-PEPS verifies the SAML Response and delivers it the CAS server for consumption (step 9). The CAS server validates the SAML Response and extracts the student's attributes to call the Academic API either to search for the ISTID via the Numero de Identificacao do Cidadao (NIC) or to create and enrol a student accessing for the first time (step 10). If everything executed correctly, the CAS server receives the ISTID of the student from the API call response and authenticates him, generating the authentication ticket. Finally, the student is redirected to Academic, that receives the ticket and validates it with the CAS server (step 11). If the validation of the ticket is successful, the student is granted access to his personal area where he can access the content of the course (step 12). The described steps can be seen in figure 4.6. Notice that most steps imply the redirection of the user browser using a RESTful approach (except for steps 10 and 11), however, the arrows that would represent the HTTP GET requests have been omitted from the figure for simplicity.

4.3 The Attribute Provider

The AP has the main purpose of serving student academic related attributes upon receiving a request from the STORK infrastructure.

The academic attributes that the AP will serve are complex attributes, meaning that each one of them has multiple values to be filled. An overview of the attributes is provided below.

generalDiploma It is a complex XML structure that is constructed following the European Diploma Supplement standard specification. It contains information regarding a diploma of the citizen.

currentStudies It is a structure very similar to the *generalDiploma*, but instead of the diploma it contains information about the current curriculum of the citizen.

hasDegree Contains information about the current qualifications of the citizen (bachelor, master or doctorate).

isStudent Asserts a student's enrolment in a specific course. This is also the attribute used to verify the attendance of students enrolled in the eLearning course.

isAcademicStaff Asserts if the citizen is part of the academic staff of the institution and what is his role (researcher or teacher).

isTeacherOf If the citizen is a professor or teaches a course, identifying it and expressing the teacher's role (professor, mentor or tutor).

isCourseCoordinator Asserts if the citizen is the coordinator of a course and identifies it.

isAdminStaff Asserts if the citizen is part of the university staff in some way (e.g.: grant owner, employee or alumni).

The values to construct these attributes are obtained using information that is stored by Fenix. Even though most of the information can be retrieved from the Fenix database with domain function calls, it's often the case that the information has to be interpreted and processed to fulfil the specification of the attributes and respective values. For example, to respond with the *isCourseCoordinator* attribute, it's necessary to first retrieve the Person object referring to the person whom NIC value was received in the attribute request. This is easily done because there's a specific domain function that allows the retrieval of a Person given its NIC. Once the Person object has been retrieved, it is necessary to check if it has some reference to a Professor object, meaning that the Person may be a professor, from there, other domain calls are used to check if some coordinator role exists for the Professorship class associated to the Professor object.

For the *generalDiploma* it gets much more complicated, since the EDS is a complex XML structure and each value of its composing fields has to be retrieved using domain functions, process the data in some way and finally, insert it into the XML tree. Because the SAML is an XML structure itself, so that the EDS doesn't cause problems when the SAML needs to be parsed, it needs to be escaped so that it can be treated as a string and inserted as the value for the *generalDiploma*. These were just some examples to explain that to fill some more complex attribute values, a lot of processing needs to occur.

4.3.1 Software Architecture

The overall architecture of the AP and the involved components can be seen in Figure 4.7.

As seen in the figure, the AP has been developed in a separate module that is injected as a dependency on the Academic platform and is included in the assembled WAR file upon compilation. This tight integration into the Academic platform was necessary because the students information stored in its database is accessible through its domain level methods. The AP endpoint is automatically available upon deployment of the Fenix platform and it is externally accessible without the requirement of authentication at the application level. However, so that the endpoint is secured, some measures have been taken to prevent the web service from being invoked by unauthorized users. This will be explained in further detail in 4.3.3.

The AP receives requests directly from the PIAP (see section 3.5) via a SOAP web service, being completely agnostic of the SAML messages and protocol, however, the attributes in the response have to be compliant to the STORK specifications.

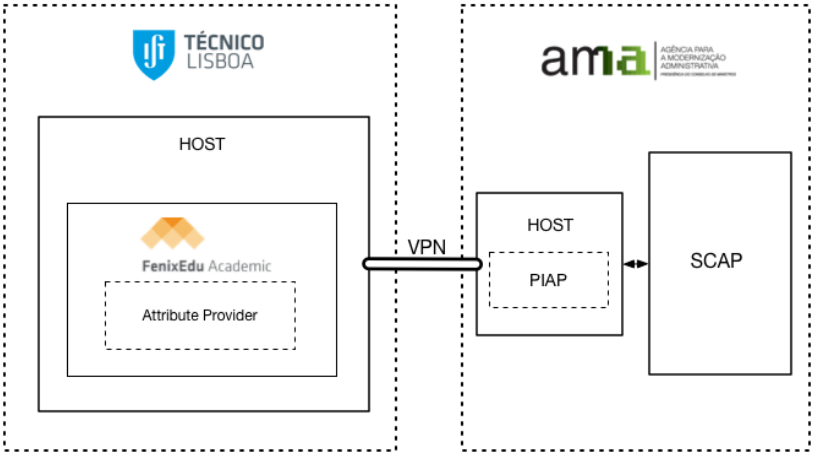


Figure 4.7: Overview of the Attribute Provider Architecture

4.3.2 Implementation

The PIAP is the Portuguese IdM interface between the STORK infrastructure and the national APs. The protocol used for the communications is the SOAP web service standard whose interface is defined by a WSDL file that specifies the format and content types of the request received by the APs and its response to the PIAP. The communication model is described in Figure 4.8 and it is asynchronous, meaning that the APs must respond to the PIAP immediately after receiving a request, acknowledging it with a HTTP "200 OK" message. After the acknowledge is sent, the request is processed and once all the attributes have been constructed, another web service provided by the PIAP is invoked from the AP, with the response in the request parameters.

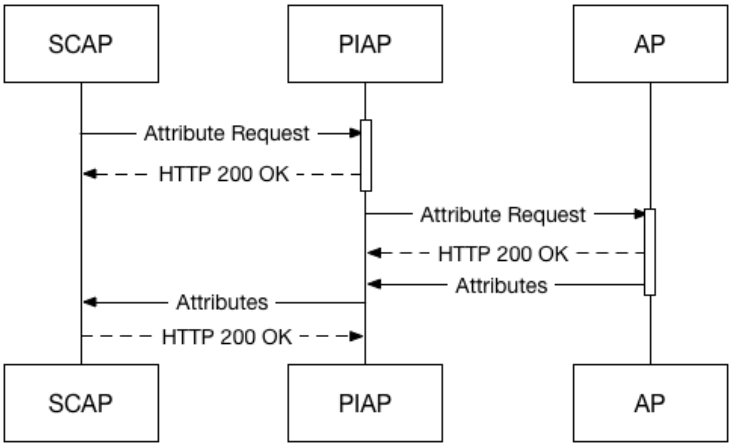


Figure 4.8: PIAP and AP interaction diagram.

The advantage of this asynchronous model is that the PIAP doesn't have to wait for each AP response individually before requesting attributes to the next AP, so that if they take a considerable amount of time to process the request and respond with the attributes, instead of synchronously waiting for the first to respond before moving to the next, it can send the requests to all the AP letting them process the requests simultaneously and respond as they finish, forwarding the responses to the SCAP for signing the attributes.

The AP web service that the PIAP invokes to get a citizen's academic attributes exposes a single

remote call with the signature: *getAttributes(SingleAttributeRequestType)*. The *SingleAttributeRequestType* is a complex type defined in one of the binding schemas that contains information regarding the citizen whose attributes are being requested that the AP can use to identify him, such as his name and NIC.

4.3.3 Security

The communications between the AP and the PIAP are secured using a Virtual Private Network (VPN) established between the machines running the web services. This way, the exchanged messages between the two systems are secured at the network layer instead of the application layer. However, because the AP is part of the Academic platform, it is publicly accessible via the Internet. This causes a security problem, because even though the VPN secures the connection with the PIAP, the web service is still publicly available. To solve this, an handler has been added to the service that filters any request that arrives at the AP endpoint and whose source IP isn't the IP address of the PIAP within the established VPN subnet. The web service is still vulnerable to IP spoofing attacks. Ideally the web service should be secured at the application layer by using the PKI to enforce access control policies, however the PIAP doesn't support this.

4.4 System overview

Now that we have explained all the components of the academic pilot, we will give a macro view of how it all links together. Figure 4.9 shows the overall components and their interactions.

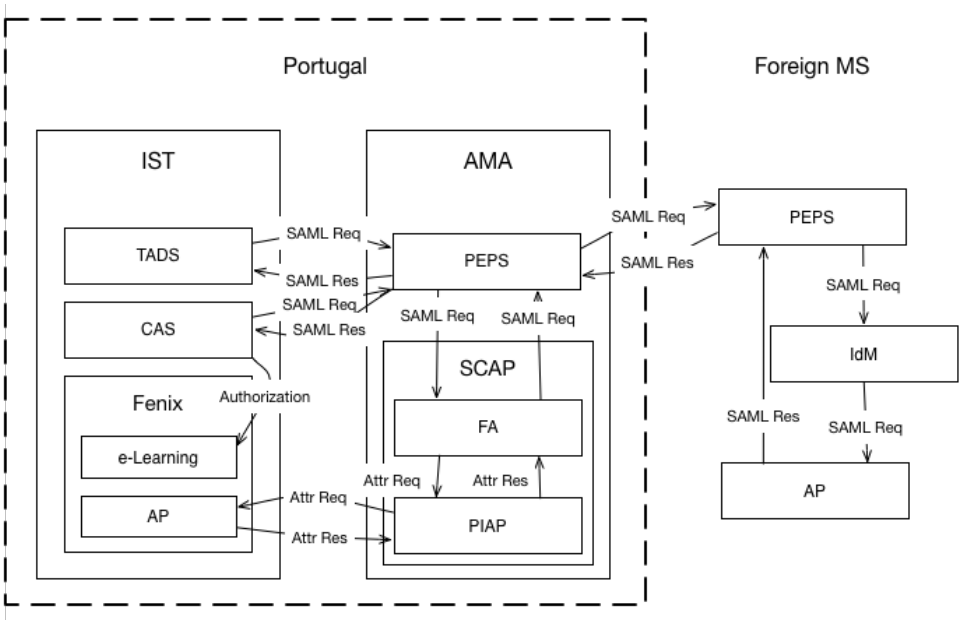


Figure 4.9: Overview of the pilot components.

Both the TADS and eLearning pilots are deployed on two virtual machines running on the same host. A STORK SAML authentication and attribute retrieval flow may be started either by an user accessing TADS or by a student trying to access the eLearning course. In both cases, the SAML requests are forwarded to the national PEPS, which is maintained by AMA. From there, depending on the country of origin the user selected before the request was created, the SAML message is either redirected to a foreign PEPS (if the user is foreign) or to the national IdM (if the user is Portuguese). If the request

was forwarded to a foreign IdM, the national PEPS just has to receive the response and send it back to the requester. Otherwise, the PEPS has to forward the request to the national IdM (SCAP). The various components of the SCAP are responsible for authenticating the user (the FA component) and retrieve the attributes (the PIAP component). The PIAP then requests the attributes to all the APs that are available (Figure 4.9 only shows IST's AP), which in turn respond with the attributes. Imagining that a request IST's AP receives requires the EDS attribute, it would serve it back to the PIAP. The PIAP then sends it back to SCAP, where the attribute will be digitally signed. Finally, the SAML response is constructed and is delivered back to the SP (TADS, in this use case example), which can process it and validate its content, using it for access control or other purposes.

4.5 Discussion

In this section we presented our solutions for the components of the STORK2.0 academic pilot in Portugal.

We described the software architectures of all the different systems and their different relevant components, as well as give an overview on how the different systems all connect together through the national IdM and STORK infrastructure. Although the systems are very heterogeneous, by using different programming languages and frameworks they were all integrated successfully with the existing software and with the infrastructure to deploy the pilot.

This demonstrates that the STORK infrastructure is agnostic from the national IdM implementations by describing the interfaces to request authentication and attributes using the SAML standard, although integration is always required, it achieves the goal of creating an abstraction over the complexities of these systems deployed at the MS's national level.

We have also illustrated the security measures that have been taken into consideration while developing the different components, given their different requirements.

5

Evaluation

Contents

This chapter presents the evaluation of the system components. Some components were tested in a pre-production environment while others only locally, with a test deployment of the PEPS infrastructure.

5.1 Component Testing

At the time of writing of this document, the pilot is still ongoing, but cross border tests haven't been possible yet due to delays in the deployment of the national STORK infrastructure. All the pilots were successfully deployed in a test environment using a local deployment of all the different components of the PEPS model to simulate the whole route the SAML messages would go through, as well as authentication and attribute retrieval. Because of this, no user tests with foreign partners were possible using the infrastructure.

5.1.1 TADS

TADS was tested in a pre-production environment. It is currently possible to get attributes from the national PEPS (the example seen in Annex D is a document retrieved using the national IdM infrastructure), but not from the developed AP provider.

The construction of the EDS (both the current studies and diploma) attribute was only possible locally with some example SAML Responses containing those attributes in the response and loading it from disk. Some partners that have already deployed TADS with their national infrastructure have reported the correct functioning of the application.

5.1.2 eLearning

The eLearning was also only tested locally and is in pre-production waiting for testing with some foreign partners. This is because the national pre-production PEPS deployment is still unreliable and has issues that have to be resolved before cross-border testing can begin.

5.1.3 AP

The attribute provider is currently connected to the PIAP and communication was successful, as reported by AMA. However, no information on the current state of the connection between the PIAP, SCAP and the STORK infrastructure have been reported.

The creation of complex attributes (such as the EDS) XML structures have been tested manually by forging a SAML Response that would contain those attributes, and then proceeding to processing it with TADS by creating a document containing them. These tests were successful, but testing with the infrastructure is still required

5.2 Security Analysis

This section will discuss some of the security measures that have been taken in order to secure the applications that were developed for the STORK 2.0 academic pilot. Some of the possible attack vectors will be listed and explained as well as the respective mitigation measures taken.

5.2.1 TADS

A number of well known web application vulnerabilities are common on the internet. We will list them and explain the measures taken to mitigate them on TADS.

5.2.1.A HTTPS

HTTPS is obligatory in TADS to protect the user's privacy and confidentiality. Not only because the users authentication credentials need to be delivered securely to the server, but also the documents that the user retrieves with his browser from TADS. Moreover, HTTPS connections are mandatory with all the involved STORK entities in all the SAML HTTP Redirect Bindings, so that the confidentiality of the SAML assertion's content is undisclosed to malicious entities.

However, HTTPS has some know vulnerabilities in the allowed key exchange protocol and used cipher suites, so it is important that the web application rejects some protocol versions or cipher suites during the algorithm negotiation phase to mitigate those attacks.

Because HTTPS relies on the PKI and public key certificates, the size of the private key of the server is important to determine the strength of the encryption. For the private key, a 2048 bit length was used, which is stated to be secure until 2030 by the RSA. However, the CA that signed the certificate used the SHA-1 algorithm to hash the certificate. This is hash function is considered unsecure, as it's been proven that it is not collision resistant and it is feasible to create different messages that output the same hash value. This means that it's theoretically possible for someone to forge a valid certificate with the same hash result that could lead to the impersonation of the TADS server.

SSL version 3.0 is susceptible to a number of different attacks such as the POODLE attack. This makes it possible for an attacker to compromise a TLS session with a man-in-the-middle attack, by relying on the fallback to SSL 3.0 in the handshake and cipher suite negotiation phase of the protocol. For this reason SSL 3.0 has been disabled on the server.

TADS uses HTTPS for every connection except the first time the browser connects to the website. Because the protocol must run on a different port, it is common practice to ensue an HTTP status code 302, redirecting to the HTTPS URL of the website every time an HTTP request is received. TADS does this, but additionally uses the HTTP Strict Transport Security (HSTS) header. This header is valid for a configurable amount of time, and it tells the browser to strictly connect using HTTPS the next time the user tries to connect to the website, minimizing the number of HTTP connections.

The security of the HTTPS connection was tested using the Qualys SSL LABS SSL Server Test[1]. And achieved a C grade. The output of the test is presented in Annex B.

5.2.1.B Cross site scripting (XSS)

XSS allows an attacker to inject JavaScript scripts into a web application by sending it untrusted data. If the website is vulnerable to the attack, the scripts can then be sent to the victim's browser when accessing the application, which will be executed by the browser, leading to possible cookie stealing or redirection of the browser to untrusted websites controlled by the attacker. This can potentially have catastrophic consequences because cookie stealing can lead an attacker to hijack any user's active SSO session.

TADS only allows the user to input a string into a single form in the verification page, where the verifier must input his email to receive the link to the document retrieval web page. This could potentially be exploited by an attacker to try to inject a script, however the submitted information into the form is never displayed back to the user, and it's only used in the back end to send the email to the verifier. If the

[1] https://www.ssllabs.com/ssltest/

string isn't a valid e-mail an error is returned to the browser, and nothing more. Either way, the submitted input of the form is validated on the server to make sure that only valid emails are passed to the SMTP client.

5.2.1.C Cross Site Request Forgery (CSRF)

A CSRF attack consists in sending a forged HTTP request to a logged in victim's browser. This is often achieved by making the victim click some forged URL that redirects them to the vulnerable web application with some HTTP parameters on the request. Another method is hosting a malicious website that has some HTML *img* tag pointing to the malicious URL, which will trigger an HTTP GET to retrieve the image from that URL. These parameters are processed by the web application as if the operation was intentionally requested by the user if he has an active session established by having the associated cookie. The outcome of the operation usually leads to some benefit for the attacker. This attack is very simple to perform because the attacker doesn't need access to any of the credentials of the user but only needs to issue a valid HTTP URL that can be successfully processed by the web application.

There are several checks that can be made in order to mitigate the CSRF attack. The first is the usage of the Synchronizer Token Pattern. This pattern consists in issuing a secret challenge when the logged in user loads a page with a HTML FORM control or any sensitive server-side operation. The Server will then expect to receive this token when the FORM is submitted, and will only proceed to executed the requested operation if the challenge is correct. This ensures that the user actually intended to submit the desired request. The token value must be randomly generated such that an attacker cannot guess it.

Other methods to mitigate the attack include the verification of the HTTP Origin and Referer headers. The server can verify if the Origin and/or referer match its domain and only accept requests that originally came from itself, mitigating the possibility of the request being forged on some malicious domain or somewhere else (in which case the headers would have a *null* value).

5.2.1.D Injection

Another common web application attack is called injection and it consists in sending untrusted that to the server that is interpreted as a database query or command. Both SQL and noSQL databases are prone to the attack. Every time some piece of data the user inputs into a form has to be validated against some database entry, the user input has to be properly validated to ensure that it's not a malicious script. An example would be the insertion of a string that when validated to SQL returns true (e.g.: "1=1"). Let's consider an username and password login form. If the query is constructed by concatenating the username and the password to the query in some way that it forms a valid statement, without validating the input, if the password is some string with an SQL "OR" statement where one of the conditions is true, it would return "true" regardless of the value of both the username or password, returning all the rows of the database table or the password hashes (or plain text if the entries aren't stored securely). In noSQL the same thing can be achieved. Another possible injection is exploiting the JavaScript eval() function. If this function is used on the server side to evaluate strings that the user inputs into some forms into actual JavaScript, it allows to run commands that will kill the server process, shutting down the service altogether, leading to a Denial of Service (DoS).

TADS doesn't accept any string input by the user because it isn't required by its business logic. Moreover, the authentication isn't handled by the server itself, but by the STORK infrastructure, so these types of attacks have to be considered by the IdM, but cannot be done on TADS.

5.2.2 eLearning

The eLearning pilot has two possible attack vectors: the Fenix platform and the CAS server. We will briefly go through some of the possible vulnerabilities of both and the measures taken to mitigate any exploits.

5.2.2.A Fenixedu-Academic

As has already been explained, Fenix relies on the CAS server for authentication. This means that it uses the CAS client to verify if an user trying to access the server has an active SSO session, and if not, it redirects the user to the CAS authentication page. Fenix uses the Apache Struts framework to implement the MVC architecture and the Spring webflow framework to control the flow of page navigation and its routes.

The webflow framework is configured so that every time an user tries to access the application, the request goes through a number of filters. These filters check if the user already have an active session in the web application (meaning that the SSO session has already been established). If not, it checks the HTTP request for a parameter called ticket (see 3.4.1). If it exists, the user already authenticated and has been redirected back to Fenix. If the filter a CAS ticket in the request, it invokes the CAS client. The client then establishes a secure connection (via TLS) with the CAS server and validates the ticket. If the ticket is valid, the client retrieves the principle associated to the user that logged in, and establishes a new session for him. Every Fenix request goes through this filter, meaning that the filter always looks for either an active session or a CAS ticker, and if none of both are found, he is not granted access to the application.

The Fenix API described in the 4.2 however, is one of the few cases where the requests don't go through the filter described above. The route to the API is accessible to anyone, but the REST calls are protected by a configured username and password tuple, so that only clients that provide the credentials may access the services provided by the API, authenticating themselves. The provided API services also include the on-demand creation and enrolment of a new STORK participant student that is trying to access the eLearning pilot by providing the necessary SAML assertions to do so. If someone compromises the password, it could potentially lead to the creation of fake students that would populate and pollute the system's database with fake information. This can lead to other problems such as a possible DoS of the database.

5.2.2.B Central Authentication Service

The CAS server relies on the HTTPS protocol to secure the delivery of the user credentials between his browser and the server. The confidentiality of the credentials that are sent over the network depend on the quality of the used cipher suite and the issued public key certificate from the CA. The same applies for the SAML assertions that the CAS server requests using the SAML HTTP Redirect Binding.

An attack to the SAML authentication at the CAS server wouldn't give the attacker any sort of privileged access to the attacker, but only access to the online course. For this reason, it is more likely that an attacker would try to gain access using the username/password login and try to access the account of a privileged user using a brute force attack or so, which would be easier than trying to compromise a SAML message (because it involves hijacking the TLS connection and also compromising the signature of the SAML assertions).

5.2.3 AP

The attribute provider is a SOAP web service attached to the Fenix web application. Because of this, the SOAP endpoint is publicly available on the internet. This means that anyone can look at the service WSDL and can try to invoke one of its web methods. Ideally, the AP would be deployed on a machine that isn't publicly available and is only connected to the PIAP (see 3.5) via the established VPN. This is not possible however, because the endpoint has to run alongside Fenix to be able to access its database and use its domain method calls to retrieve the user's information.

Because AMA has established that the communication is secured via the VPN, a handler is responsible to do access control by verifying if the request came from an IP address that is part of the subnet of the VPN. This is however insecure, because IP spoofing is relatively easy, which could lead to disclosure of sensitive data about some user since the only information required to get the attributes described in 4.3 of a user is his NIC. The best solution to this problem would be to use authentication of the client at the web service level using public key certificates instead of relying o the handler. This was proposed to AMA but no response was obtained.

5.2.4 The STORK Infrastructure

The STORK entities must also take into consideration the possible attack vectors described above. However, another attack is possible due to the XML nature of the SAML messages, called an XML External Entity processing attack and will be described below.

5.2.4.A XML External Entity (XXE) Processing

The XXE attack takes advantage of the XML standard against any application that parses XML. This includes the openSAML libraries that are used in the STORK SAMLEngine and other project libraries to parse the SAML messages (which are essentially XML files). This vulnerability is critical because it may lead to the disclosure of sensitive data.

The XML standard defines a concept called an entity, which is basically data that can be referred internally to the XML document (to avoid repetition) or can be found externally to the XML document being parsed. The latter is the definition that causes the vulnerability. An XML Entity of the type External can be either a SYSTEM or PUBLIC entity, both identified by an URI. A SYSTEM external entity is particularly dangerous because it allows the client of the parser to send it an XML document containing one or more External SYSTEM entities identified by URI's that can locate resources on the file-system of the server. When the parser encounters such an entity within the document, it will locate the resource identified by the URI and simply dump the content of the resource as the value of the tag in the document. This means that if the user sends some XML entity like exemplified in the snippet 5.1, the parser will locate the system resource and replace the content of the *bar* tag with the content of the "/etc/passwd" file. Guessing or knowing the filesystem structure it is possible for the client of the parser to get the content of any file that is readable by the process running the parser. The attacker can access these values by making a deliberate mistake later on the XML structure, making the parser return the content of the XML and the error back to the client. This can lead to the disclosure of private keys to the client.

The XEE attack can also lead to DoS of the service. If the URI of the system resource being accessed is either "/dev/null" or "/dev/random", the parser will infinitely read either infinite *null* values or random numbers respectively, which can exhaust the server computational resources. Other ways of doing this is to define nested references to build a memory bomb.

Listing 5.1: XML XXE Example

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE foo [
  <!ENTITY bar SYSTEM "file:///etc/passwd">
]>
<someTag>&bar;</someTag>
```

Because the STORK SAML messages are signed, before the PEPS XML parser even begins to parse it, the signature is validated first. This means that the SAML Request is only accepted (and consequently parsed and processed) if it comes from a trusted partner whose public key is in the trusted keystore of the SAMLEngine of the server. This makes the vulnerability exploitable only by parters of the project, however the existence of the vulnerability might open some other attack vectors that can be exploited by any internet user and not only an LSP partner.

6

Conclusions and Future Work

Contents

In this chapter we review the work done and the achieved results, and propose possible improvements for future work.

6.1 Conclusions

This work proposed and developed software solutions for the Portuguese STORK Academic LSP, which consisted in the integration of both the developed (TADS) SP and the pre-existing Fenix SPs with the STORK infrastructure. The development of a SP from square one and the integration with the infrastructure of previously existing systems (Fenix and CAS) presented different challenges. The varying nature of the applications and technologies used demonstrates the difficult interoperability problem in creating such a large scale infrastructure. STORK tries to minimize these efforts but they are still significant as SPs not only require integration with the project's provided libraries to connect to the infrastructure but may also need additional integration to align with the business processes of the SP. Moreover, the management of the necessary PKI components is still far from ideal. Besides the technological issues, legal and political problems may arise from exchanging attributes freely across different MSs, which may need the revision of legislation and directives to protect the all the stakeholders of the federation by enabling accountability and liability in case of misuse or fraud.

These challenges allowed for a better understanding of the implication of such an infrastructure at an European level and the effort required for SPs and APs to embrace it, as well as the difficulties faced into its integration with already existing IdM solutions at a national level. As we were overcoming these challenges, useful information was gathered for the pilot and its partners.

We succeeded in the implementation of all the requirements and created the necessary conditions for user testing of the national STORK infrastructure. It is clear however, that from the gathered experience and information, while achievable, full interoperability is still far from being optimal, as usability is still poor from the users perspective and it's somewhat difficult to teach them the benefits of strong authentication, as well as how the STORK usage flow protects their data from misuse and disclosure to third parties.

6.2 Future Work

At the time of writing of this document, the LSP is ongoing and the infrastructure is still being tested. Some changes and decisions are still being made and there's still a lot of room for improvement. An example would be the provided libraries for SPs, that could use some simplification for their configuration and would benefit by offering a better and simpler interface for the applications that have to integrate them. Multi-domain SSO using the SAML messages and corresponding multi-domain logout are also still not fully implemented and require further testing.

Some solutions are also being proposed for the aggregation of attributes from different sources, a relevant problem that still needs to be solved.

The PKI and Public Key Certificates, while being a key asset for the establishment of the trust networks between the STORK partners, is also a cause of problems because of its difficult management. A solution for facilitating the exchange and update of certificates between the partners could reveal useful.

Some interesting features are still being tested such as an anonymity layer similar to the The Onion Routing (TOR) project but using the PEPSs proxies, opening up many possibilities for the development of privacy-keeping applications for the citizens of the EU.

The national IdM system interface provided to APs could use some big improvements, as its current design makes integration difficult and its SOAP API needs a complete redesign, as in its current state

it goes against the EU directive that specifically states that only the minimum amount of necessary attributes of a citizen's must be requested at the APs.

Finally, TADS could improve its security further by cyphering the documents generated by its users on the disk. The commonly used solutions to do so rely on the user's password as a key, but in the case of STORK, the password is managed outside of the domain of the SPs, making these solutions unsuitable. A possible solution could rely in using their ECC public keys when available. Some more improvements could be better cookie and session management with the help of expiring sessions and automatic deletion of documents after some preset amount of time or after their digital signature expires.

Bibliography

[1] J. Hughes and E. Maler, "Security assertion markup language (saml) v2. 0 technical overview," *OASIS SSTC Working Draft sstc-saml-tech-overview-2.0-draft-08*, pp. 29–38, 2005.

[2] D. W. Chadwick and G. Inman, "The trusted attribute aggregation service (taas)," in *2013 International Conference on Availability, Reliability and Security*. IEEE, Sep. 2013, pp. 285–290.

[3] G. López, Ó. C. Reverte, A. F. Gómez-Skarmeta, and J. Girao, "A swift take on identity management." *IEEE Computer*, vol. 42, no. 5, pp. 58–65, 2009.

[4] M. Barisch, E. T. Garcia, M. Lischka, R. Marques, R. Marx, A. Matos, A. Mendez, and D. Scheuermann, "Security and privacy enablers for future identity management systems," in *Future Network and Mobile Summit, 2010*. IEEE, 2010, pp. 1–10.

[5] J. Camenisch, D. Sommer, S. Fischer-Hübner, M. Hansen, H. Krasemann, G. Lacoste, R. Leenes, J. Tseng *et al.*, "Privacy and identity management for everyone," in *Proceedings of the 2005 workshop on Digital identity management*. ACM, 2005, pp. 20–27.

[6] B. Zwattendorfer, I. Sumelong, and H. Leitold, "Middleware architecture for cross-border eid," in *2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN)*. IEEE, 2012, pp. 303–308.

[7] T. Rössler, "Giving an interoperable e-id solution: Using foreign e-ids in austrian e-government," *Computer Law & Security Review*, vol. 24, no. 5, pp. 447–453, 2008.

[8] H. Leitold and A. Tauber, "A systematic approach to legal identity management–best practice austria," in *Proceedings of the Information Security Solutions Europe 2011 Conference*, 2011, pp. 224–234.

[9] G. Moniava, E. Verheul, and L. Schoenmakers, "Extending digid to the private sector (digid-2)," Ph.D. dissertation, Master's thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, 2008.

[10] B. Zwattendorfer, A. Tauber, and T. Zefferer, "A privacy-preserving eid based single sign-on solution," in *Network and System Security (NSS), 2011 5th International Conference on*. IEEE, 2011, pp. 295–299.

[11] T. Martens, "Electronic identity management in estonia between market and state governance," *Identity in the Information Society*, vol. 3, no. 1, pp. 213–233, 2010.

[12] A. Kalja, J. Pold, T. Robal, and U. Vallner, "Modernization of the e-government in estonia," in *Technology Management in the Energy Smart World (PICMET), 2011 Proceedings of PICMET'11:*. IEEE, 2011, pp. 1–7.

[13] T. Rissanen, “Electronic identity in finland: Id cards vs. bank ids,” *Identity in the Information Society*, vol. 3, no. 1, pp. 175–194, 2010.

[14] B. P. Bruegger, D. Hühnlein, and M. Kreutzer, “Towards global eid-interoperability.” in *BIOSIG*, 2007, pp. 127–140.

[15] J. Lopez, R. Oppliger, and G. Pernul, “Authentication and authorization infrastructures (aais): a comparative survey,” *Computers & Security*, vol. 23, no. 7, pp. 578–590, 2004.

[16] R. McKenzie, M. Crompton, and C. Wallis, “Use cases for identity management in e-government,” *IEEE Security & Privacy*, vol. 6, no. 2, pp. 51–57, 2008.

[17] N. Klingenstein, “Attribute aggregation and federated identity,” in *Applications and the Internet Workshops, 2007. SAINT Workshops 2007. International Symposium on*. IEEE, 2007, pp. 26–26.

[18] E. Bertino and K. Takahashi, *Identity Management: Concepts, Technologies, and Systems*. Artech House, 2010.

[19] S. S. Shim, G. Bhalla, and V. Pendyala, “Federated identity management,” *Computer*, vol. 38, no. 12, pp. 120–122, 2005.

[20] J. Jensen, “Federated identity management challenges,” in *Availability, Reliability and Security (ARES), 2012 Seventh International Conference on*. IEEE, 2012, pp. 230–235.

[21] M. I. Chehab and A. E. Abdallah, “Architectures for identity management,” in *Internet Technology and Secured Transactions, 2009. ICITST 2009. International Conference for*. IEEE, 2009, pp. 1–8.

[22] R. Dhamija and L. Dusseault, “The seven flaws of identity management: Usability and security challenges,” *Security & Privacy, IEEE*, vol. 6, no. 2, pp. 24–29, 2008.

[23] A. Jøsang and S. Pope, “User centric identity management,” in *AusCERT Asia Pacific Information Technology Security Conference*, 2005, p. 77.

[24] J. Torres, M. Nogueira, and G. Pujolle, “A survey on identity management for the future network,” *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 2, pp. 787–802, 2013.

[25] I. Jorstad, T. A. Johansen, E. Bakken, C. Eliasson, M. Fiedler, and v. T. Do, “Releasing the potential of openid & sim,” in *13th International Conference on Intelligence in Next Generation Networks*, no. 2624. IEEE, Oct. 2009, pp. 1–6.

[26] S. Arora, “National e-id card schemes: A european overview,” *Information Security Technical Report*, vol. 13, no. 2, pp. 46–53, 2008.

[27] A. Hisamatsu, D. Pishva, and G. Nishantha, “Online banking and modern approaches toward its enhanced security,” in *Advanced Communication Technology (ICACT), 2010 The 12th International Conference on*, vol. 2. IEEE, 2010, pp. 1459–1463.

[28] R. T. Fielding and R. N. Taylor, “Principled design of the modern web architecture,” *ACM Transactions on Internet Technology (TOIT)*, vol. 2, no. 2, pp. 115–150, 2002.

[29] R. Morgan, S. Cantor, S. Carmody, W. Hoehn, and K. Klingenstein, “Federated security: The shibboleth approach.” *Educause Quarterly*, vol. 27, no. 4, pp. 12–17, 2004.

[30] R. Oppliger, S. Gajek, and R. Hauser, “Security of microsoft’s identity metasystem and cardspace,” in *Communication in Distributed Systems (KiVS), 2007 ITG-GI Conference*. VDE, 2007, pp. 1–12.

[31] D. W. Chadwick and G. Inman, “Attribute aggregation in federated identity management.” *IEEE Computer*, vol. 42, no. 5, pp. 33–40, 2009.

[32] W. Priesnitz Filho and C. Ribeiro, “Obtaining strong identifiers through attribute aggregation,” *SECURWARE 2014*, p. 107, 2014.

[33] R. Oppliger, “Computing practices - Microsoft .NET passport: a security analysis,” *Computer*, vol. 36, no. 7, pp. 29–35, Jul. 2003.

[34] ——, “Microsoft. net passport and identity management,” *Information Security Technical Report*, vol. 9, no. 1, pp. 26–34, 2004.

[35] T. El Maliki and J.-M. Seigneur, “A survey of user-centric identity management technologies,” in *Emerging Security Information, Systems, and Technologies, 2007. SecureWare 2007. The International Conference on*. IEEE, 2007, pp. 12–17.

[36] P. Riesch and X. Du, “Audit based privacy preservation for the openid authentication protocol,” in *Homeland Security (HST), 2012 IEEE Conference on Technologies for*, 2012, pp. 348–352.

[37] J.-B. Lezoray and M. Pasquet, “Enabling collaboration between heterogeneous circles of trust through innovative identity solutions,” in *Collaborative Technologies and Systems, 2009. CTS’09. International Symposium on*. IEEE, 2009, pp. 476–484.

[38] M. Ates, J. Fayolle, C. Gravier, and J. Lardon, “Complex federation architectures: stakes, tricks & issues,” in *Proceedings of the 5th international conference on Soft computing as transdisciplinary science and technology*. ACM, 2008, pp. 152–157.

[39] A. Pérez, G. López, Ó. Cánovas, and A. F. Gómez-Skarmeta, “Formal description of the swift identity management framework,” *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1113–1123, 2011.

[40] A. Matos, J. Girão, S. Sargento, and R. Aguiar, “Preserving privacy in mobile environments with virtual network stacks,” in *Global Telecommunications Conference, 2007. GLOBECOM’07. IEEE*. IEEE, 2007, pp. 1971–1976.

[41] S. S. Garcia, A. Gomez Oliva, E. P. Belleboni, and I. Pau De La Cruz, “Current trends in pan-european identity management systems,” *Technology and Society Magazine, IEEE*, vol. 31, no. 3, pp. 44–50, 2012.

[42] C. Andersson, J. Camenisch, S. Crane, S. Fischer-Hubner, R. Leenes, S. Pearsorr, J. S. Pettersson, and D. Sommer, “Trust in prime,” in *Signal Processing and Information Technology, 2005. Proceedings of the Fifth IEEE International Symposium on*. IEEE, 2005, pp. 552–559.

[43] V. Koulolias, A. Kountzeris, H. Leitold, B. Zwattendorfer, A. Crespo, and M. Stern, “Stork e-privacy and security,” in *5th International Conference on Network and System Security*. IEEE, Sep. 2011, pp. 234–238.

[44] C. Adams and S. Lloyd, *Understanding PKI: concepts, standards, and deployment considerations*. Addison-Wesley Professional, 2003.

[45] P. Aubry, V. Mathieu, and J. Marchal, “Esup-portail: open source single sign-on with cas (central authentication service),” *Proc. of EUNIS04–IT Innovation in a Changing World*, pp. 172–178, 2004.

A

Annex 1

This Annex describes some of the User Interface (UI) for TADS and the CAS STORK integration.

A.1 User Interfaces

A.1.1 TADS

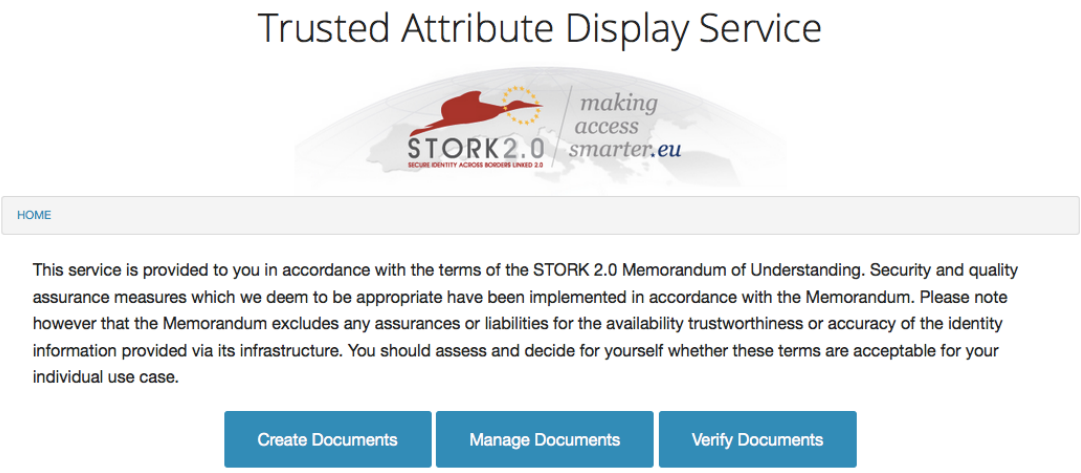


Figure A.1: The TADS landing page where the user is presented with the options for document creation, management and verification.

A.1.2 CAS

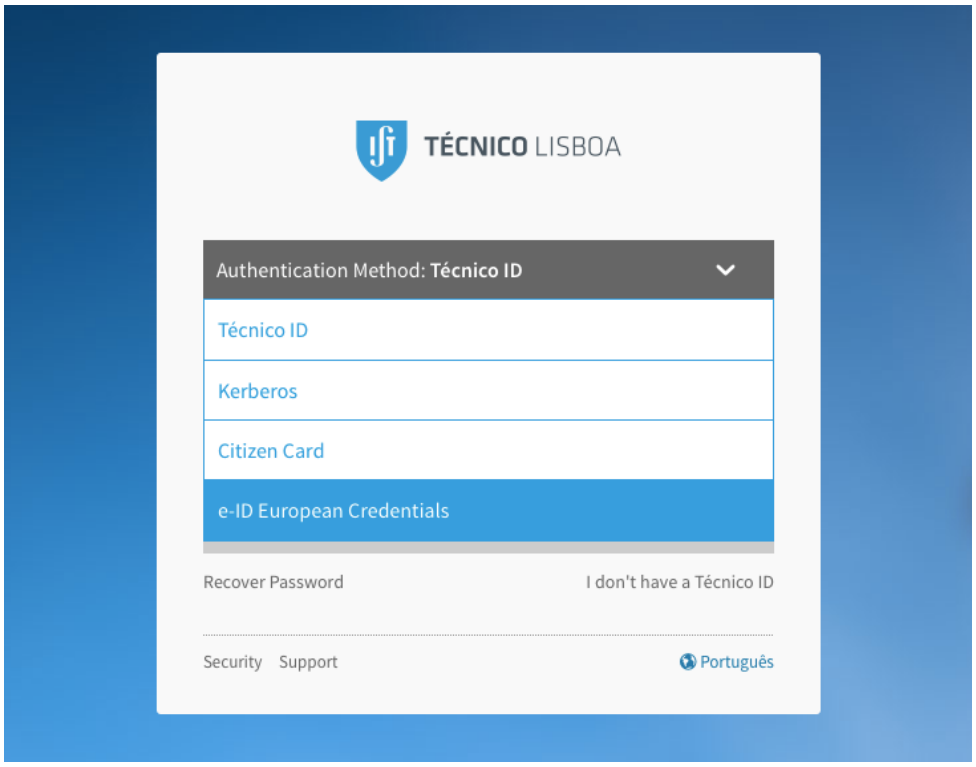


Figure A.2: The CAS drop-down menu where the user can select the “e-ID European Credentials” as an authentication option.

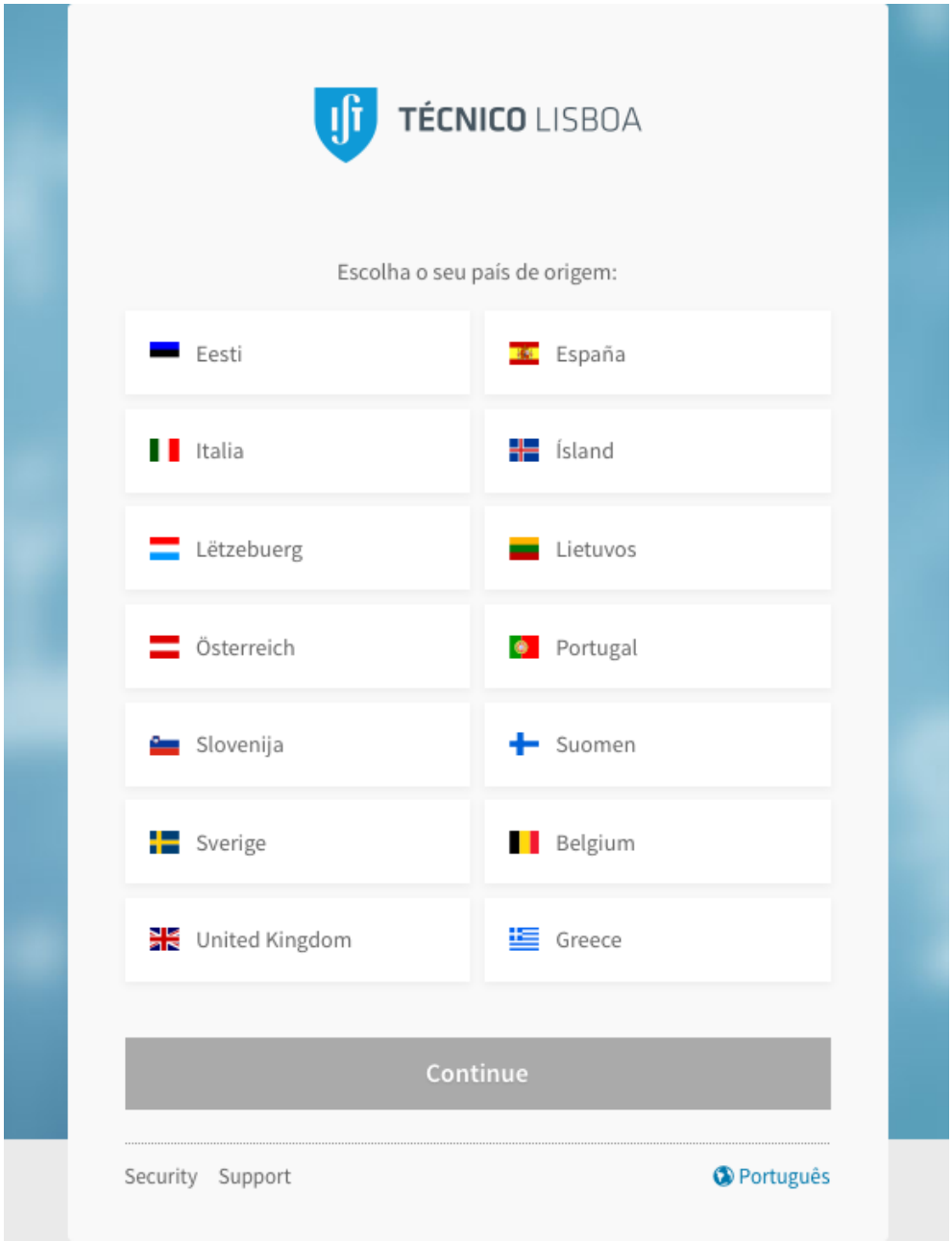


Figure A.3: The CAS UI where the user can select his country of origin for correct redirection to the C-PEPS.

B

Annex 2

B.1 SSL TADS Test

The results below assess the security of the HTTPS connection to the TADS server.

QUALYS SSL LABS

Home Projects Qualys.com Contact

You are here: Home > Projects > SSL Server Test > tads.tecnico.ulisboa.pt

SSL Report: tads.tecnico.ulisboa.pt (193.136.160.13)

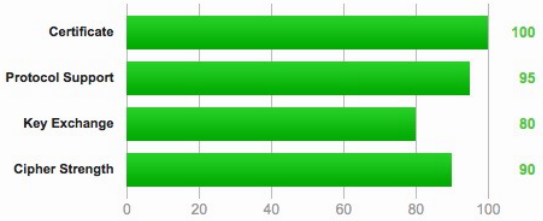
Assessed on: Sat Apr 25 08:00:40 PDT 2015 | Clear cache

Scan Another »

Summary

Overall Rating

B



Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

Certificate has a weak signature and expires after 2016. Upgrade to SHA2 to avoid browser warnings. MORE INFO »

This server accepts the RC4 cipher, which is weak. Grade capped to B. MORE INFO »

The server does not support Forward Secrecy with the reference browsers. MORE INFO »

Authentication

Server Key and Certificate #1

Common names	tads.tecnico.ulisboa.pt
Alternative names	tads.tecnico.ulisboa.pt tads-quality.tecnico.ulisboa.pt
Prefix handling	Not required for subdomains
Valid from	Wed Jun 11 17:00:00 PDT 2014
Valid until	Sun Jun 11 16:59:59 PDT 2017 (expires in 2 years and 1 month)
Key	RSA 2048 bits (e 65537)
Weak key (Debian)	No
Issuer	TERENA SSL CA
Signature algorithm	SHA1withRSA WEAK
Extended Validation	No
Revocation information	CRL, OCSP
Revocation status	Good (not revoked)
Trusted	Yes

Additional Certificates (if supplied)

Certificates provided	4 (4535 bytes)
Chain issues	Contains anchor

#2

Subject	TERENA SSL CA Fingerprint: 3a881764472b6441ddb3afdd47c6b8b76ee7ba1d
Valid until	Sat May 30 03:48:38 PDT 2020 (expires in 5 years and 1 month)
Key	RSA 2048 bits (e 65537)
Issuer	UTN-USERFirst-Hardware

Signature algorithm	SHA1withRSA **WEAK**

#3

Subject	UTN-USERFirst-Hardware Fingerprint: 3d4b2a4c64317143f50258d7e6fd7d3c021a529e
Valid until	Sat May 30 03:48:38 PDT 2020 (expires in 5 years and 1 month)
Key	RSA 2048 bits (e 65537)
Issuer	AddTrust External CA Root
Signature algorithm	SHA1withRSA **WEAK**

#4

Subject	AddTrust External CA Root In trust store Fingerprint: 02faf3e291435468607857694df5e45b68851868
Valid until	Sat May 30 03:48:38 PDT 2020 (expires in 5 years and 1 month)
Key	RSA 2048 bits (e 65537)
Issuer	AddTrust External CA Root Self-signed
Signature algorithm	SHA1withRSA Weak, but no impact on root certificate

Certification Paths

Path #1: Trusted

1	Sent by server	tads.tecnico.ulisboa.pt Fingerprint: 03da282e8fe1ff7bee3d19545185408521e75855 RSA 2048 bits (e 65537) / SHA1withRSA **WEAK SIGNATURE**
2	Sent by server	TERENA SSL CA Fingerprint: 3a881764472b6441ddb3afdd47c6b8b76ee7ba1d RSA 2048 bits (e 65537) / SHA1withRSA **WEAK SIGNATURE**
3	In trust store	UTN-USERFirst-Hardware Self-signed Fingerprint: 0483ed3399ac3608058722edbc5e4600e3bef9d7 RSA 2048 bits (e 65537) / SHA1withRSA Weak or insecure signature, but no impact on root certificate

Path #2: Trusted

1	Sent by server	tads.tecnico.ulisboa.pt Fingerprint: 03da282e8fe1ff7bee3d19545185408521e75855 RSA 2048 bits (e 65537) / SHA1withRSA **WEAK SIGNATURE**
2	Sent by server	TERENA SSL CA Fingerprint: 3a881764472b6441ddb3afdd47c6b8b76ee7ba1d RSA 2048 bits (e 65537) / SHA1withRSA **WEAK SIGNATURE**
3	Sent by server	UTN-USERFirst-Hardware Fingerprint: 3d4b2a4c64317143f50258d7e6fd7d3c021a529e RSA 2048 bits (e 65537) / SHA1withRSA **WEAK SIGNATURE**
4	Sent by server In trust store	AddTrust External CA Root Self-signed Fingerprint: 02faf3e291435468607857694df5e45b68851868 RSA 2048 bits (e 65537) / SHA1withRSA Weak or insecure signature, but no impact on root certificate

Configuration

Protocols

TLS 1.2	Yes
TLS 1.1	Yes
TLS 1.0	Yes
SSL 3	No
SSL 2	No

Cipher Suites (sorted by strength; the server has no preference)

TLS_RSA_WITH_RC4_128_SHA (0x5) **WEAK**	128
TLS_RSA_WITH_AES_128_CBC_SHA (0x2f)	128
TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x33) DH 1024 bits (p: 128, g: 1, Ys: 128) FS **WEAK**	128
TLS_RSA_WITH_CAMELLIA_128_CBC_SHA (0x41)	128
TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA (0x45) DH 1024 bits (p: 128, g: 1, Ys: 128) FS **WEAK**	128
TLS_RSA_WITH_SEED_CBC_SHA (0x96)	128
TLS_DHE_RSA_WITH_SEED_CBC_SHA (0x9a) DH 1024 bits (p: 128, g: 1, Ys: 128) FS **WEAK**	128
TLS_RSA_WITH_AES_128_CBC_SHA256 (0x3c)	128
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 (0x67) DH 1024 bits (p: 128, g: 1, Ys: 128) FS **WEAK**	128
TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c)	128
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x9e) DH 1024 bits (p: 128, g: 1, Ys: 128) FS **WEAK**	128
TLS_RSA_WITH_3DES_EDE_CBC_SHA (0xa)	112
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (0x16) DH 1024 bits (p: 128, g: 1, Ys: 128) FS **WEAK**	112
TLS_RSA_WITH_AES_256_CBC_SHA (0x35)	256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39) DH 1024 bits (p: 128, g: 1, Ys: 128) FS **WEAK**	256
TLS_RSA_WITH_CAMELLIA_256_CBC_SHA (0x84)	256
TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x88) DH 1024 bits (p: 128, g: 1, Ys: 128) FS **WEAK**	256
TLS_RSA_WITH_AES_256_CBC_SHA256 (0x3d)	256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (0x6b) DH 1024 bits (p: 128, g: 1, Ys: 128) FS **WEAK**	256
TLS_RSA_WITH_AES_256_GCM_SHA384 (0x9d)	256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x9f) DH 1024 bits (p: 128, g: 1, Ys: 128) FS **WEAK**	256

Handshake Simulation

Android 2.3.7 No SNI [2]	TLS 1.0	TLS_RSA_WITH_RC4_128_SHA (0x5) No FS RC4	128
Android 4.0.4	TLS 1.0	TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39) FS	256
Android 4.1.1	TLS 1.0	TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39) FS	256
Android 4.2.2	TLS 1.0	TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39) FS	256
Android 4.3	TLS 1.0	TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39) FS	256
Android 4.4.2	TLS 1.2	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x9f) FS	256
Android 5.0.0	TLS 1.2	TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39) FS	256
Baidu Jan 2015	TLS 1.0	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x88) FS	256
BingPreview Jan 2015	TLS 1.2	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x9f) FS	256
Chrome 40 / OS X R	TLS 1.2	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x9e) FS	128
Firefox 31.3.0 ESR / Win 7	TLS 1.2	TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x33) FS	128
Firefox 35 / OS X R	TLS 1.2	TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x33) FS	128
Googlebot Feb 2015	TLS 1.2	TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c) No FS	128
IE 6 / XP No FS [1] No SNI [2]	Protocol or cipher suite mismatch		Fail[3]
IE 7 / Vista	TLS 1.0	TLS_RSA_WITH_AES_128_CBC_SHA (0x2f) No FS	128
IE 8 / XP No FS [1] No SNI [2]	TLS 1.0	TLS_RSA_WITH_RC4_128_SHA (0x5) No FS RC4	128
IE 8-10 / Win 7 R	TLS 1.0	TLS_RSA_WITH_AES_128_CBC_SHA (0x2f) No FS	128
IE 11 / Win 7 R	TLS 1.2	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x9f) FS	256
IE 11 / Win 8.1 R	TLS 1.2	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x9f) FS	256
IE Mobile 10 / Win Phone 8.0	TLS 1.0	TLS_RSA_WITH_AES_128_CBC_SHA (0x2f) No FS	128
IE Mobile 11 / Win Phone 8.1	TLS 1.2	TLS_RSA_WITH_AES_128_CBC_SHA256 (0x3c) No FS	128
Java 6u45 No SNI [2]	TLS 1.0	TLS_RSA_WITH_RC4_128_SHA (0x5) No FS RC4	128
Java 7u25	TLS 1.0	TLS_RSA_WITH_AES_128_CBC_SHA (0x2f) No FS	128
Java 8u31	TLS 1.2	TLS_RSA_WITH_AES_128_CBC_SHA256 (0x3c) No FS	128
OpenSSL 0.9.8y	TLS 1.0	TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39) FS	256
OpenSSL 1.0.1l R	TLS 1.2	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x9f) FS	256
OpenSSL 1.0.2 R	TLS 1.2	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x9f) FS	256
Safari 5.1.9 / OS X 10.6.8	TLS 1.0	TLS_RSA_WITH_AES_128_CBC_SHA (0x2f) No FS	128
Safari 6 / iOS 6.0.1 R	TLS 1.2	TLS_RSA_WITH_AES_256_CBC_SHA256 (0x3d) No FS	256

Safari 6.0.4 / OS X 10.8.4 R	TLS 1.0	TLS_RSA_WITH_AES_128_CBC_SHA (0x2f) No FS	128
Safari 7 / iOS 7.1 R	TLS 1.2	TLS_RSA_WITH_AES_256_CBC_SHA256 (0x3d) No FS	256
Safari 7 / OS X 10.9 R	TLS 1.2	TLS_RSA_WITH_AES_256_CBC_SHA256 (0x3d) No FS	256
Safari 8 / iOS 8.1.2 R	TLS 1.2	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (0x6b) FS	256
Safari 8 / OS X 10.10 R	TLS 1.2	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (0x6b) FS	256
Yahoo Slurp Jan 2015	TLS 1.2	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x9f) FS	256
YandexBot Jan 2015	TLS 1.2	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x9f) FS	256

(1) Clients that do not support Forward Secrecy (FS) are excluded when determining support for it.
(2) No support for virtual SSL hosting (SNI). Connects to the default site if the server uses SNI.
(3) Only first connection attempt simulated. Browsers tend to retry with a lower protocol version.
(R) Denotes a reference browser or client, with which we expect better effective security.
(All) We use defaults, but some platforms do not use their best protocols and features (e.g., Java 6 & 7, older IE).

Protocol Details

Secure Renegotiation	**Supported**
Secure Client-Initiated Renegotiation	No
Insecure Client-Initiated Renegotiation	No
BEAST attack	Not mitigated server-side (more info) TLS 1.0: 0x2f
POODLE (SSLv3)	No, SSL 3 not supported (more info)
POODLE (TLS)	No (more info)
Downgrade attack prevention	**No, TLS_FALLBACK_SCSV not supported** (more info)
TLS compression	No
RC4	**Yes WEAK** (more info)
Heartbeat (extension)	Yes
Heartbleed (vulnerability)	No (more info)
OpenSSL CCS vuln. (CVE-2014-0224)	No (more info)
Forward Secrecy	**With some browsers** (more info)
Next Protocol Negotiation (NPN)	No
Session resumption (caching)	Yes
Session resumption (tickets)	Yes
OCSP stapling	No
Strict Transport Security (HSTS)	No
Public Key Pinning (HPKP)	No
Long handshake intolerance	No
TLS extension intolerance	No
TLS version intolerance	TLS 2.98
SSL 2 handshake compatibility	Yes

Miscellaneous

Test date	Sat Apr 25 07:58:29 PDT 2015
Test duration	131.124 seconds
HTTP status code	200
HTTP server signature	-
Server hostname	tads.ist.utl.pt

SSL Report v1.15.1

C

Annex 3

C.1 SAML Messages

In this Annex we provide examples of a SAML Request (listing D.1) and Response (listing D.2). The SAML Request contains an authentication request and queries for the eIdentifier, givenName and surname attributes values, and the response provides them.

Listing C.1: A SAML Request message example. label


```
<saml2p:AuthnRequest xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
                     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
                     xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
                     xmlns:stork="urn:eu:stork:names:tc:STORK:1.0:assertion"
                     xmlns:storkp="urn:eu:stork:names:tc:STORK:1.0:protocol"
                     AssertionConsumerServiceURL="https://localhost:3000/createdocument/samlreturn"
                     Consent="urn:oasis:names:tc:SAML:2.0:consent:unspecified"
                     Destination="https://localhost/PEPS/ServiceProvider"
                     ForceAuthn="true"
                     ID="_de24b22e6e6d196c6b7b085608168f5c"
                     IsPassive="false"
                     IssueInstant="2015-05-08T17:35:44.998Z"
                     ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
                     ProviderName="DEMO-SP"
                     Version="2.0"
                     >
    <saml2:Issuer
        Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">localhost</saml2:Issuer>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
            <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
            <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" />
            <ds:Reference URI="#_de24b22e6e6d196c6b7b085608168f5c">
                <ds:Transforms>
                    <ds:Transform
                        Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
                    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
                </ds:Transforms>
                <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
                <ds:DigestValue>HqQmV0GgFrxieqeBh7XuRtcS4+c=</ds:DigestValue>
            </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>
                        g6qvTJpYDvrHP+AaX8a09PgD/yXXDisHhxEoeZrX/TDLFce6C6Qz2yT0BWz7DJBzdT0zDbMS0D95
                        AtdW14SlxouB2uXhYlUNviYBQfo15Xho0JRgfWymWoVpqBRz5wwsSbNshloJGNN8N9QcGgkg6Svj
                        Ctwdmvx3a6+GobS1BVauOfNBlxCyuusBhWJ7XuPOqS70unRitkQ+j8A0cXLx5ofACXNt0cowx9Xa
                        BWIuZsWNN1TkGPXr7YJJxwsCsarAXl9B0apXoRQd4PuSfLWgly4kRVSuWzVOXhmLLinyyZjp8/Ya
                        3CKoD38HSb50WaSR5tM7/HEVg9ekJQMog6uBTg==</ds:SignatureValue>
        <ds:KeyInfo>
            <ds:X509Data>
                <ds:X509Certificate>
                        MIIDIjCCAgqgAwIBAgIES6idsjANBgkqhkiG9w0BAQUFADBTMQswCQYDVQQGEwJFUzEOMAwGA1UE
                        CAwFU3BhaW4xDzANBgNVBAcMBk1hZHJpZDEOMAwGA1UECgwFSW5kcmExEzARBgNVBAMMCmxvY2Fs
```

```
                    LWRlbW8wHhcNMTMwODI4MTY0NzM1WhcNMTUwODI4MTY0NzM1WjBTMQswCQYDVQQGEwJFUzEOMAwG
                    A1UECAwFU3BhaW4xDzANBgNVBAcMBk1hZHJpZDEOMAwGA1UECgwFSW5kcmExEzARBgNVBAMMCmxv
                    Y2FsLWRlbW8wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCx+7zEQUbt8Ot9dByR5TuY
                    NPmTwaKhECtnkxYAx9tl+Ga3kFiv+dOSeDoQOhxkICG1ZMaG0Gtk0EQWjscC1cLwDXpA0dhOm5xh
                    ndFHxa3V3Y3zomFZA7A3nwP8wt17wDrc1xi2Ye8VrwO7Vjb/F8Rgutlcrvq+LF4g9nQLoRiKUq8w
                    NFvDlvhBbLr8hMVBMY/jY7tSQ85qLqKUkWRcoDttJjgWZoO1vpBS4m+FywtYdOoHnN6fv4Dvf1r2
                    rNLgebgBvSfwznxiulwW6FBLasYX0rYKP7RfwG8TJ+FumOgAjADj2LdwK25RZNg44XU2V1z1Fp37
                    fNXhfo08LpdD1ueFAgMBAAEwDQYJKoZIhvcNAQEFBQADggEBADMRGTWOmadZyCoJNluV+M7LopjC
                    9WpFUj76X0cAoNXmt12zYqYe8hjRmBr6l2V/FXT3/TyWET1nILG46zwwAxslw3b2LvkhU+9QcuB8
                    yC8TCJJ0kgsTZ/EUZPSbwUG7Nn2jkMiyjlyKcjSCQOzEKQyR1xYwyZG40+BPeGbYjmRgm6kcmCxY
                    USWoPwNyfke9gNT80f0pUj7wJ9YwWbTIz1rhf/h7rxoMYypXo+JXwaXW/Ra8v1uDcwfKpE5ZgwAU
                    nubLXF4A+H7/N7ZvB5XDwJ4W+99nuPsKfwacD8m1sflTXcEPzzh0q6iQ9anJT94/pMctnp827zXA
                    y66jvDrin5I=
                </ds:X509Certificate>
            </ds:X509Data>
        </ds:KeyInfo>
    </ds:Signature>
    <saml2p:Extensions>
        <stork:QualityAuthenticationAssuranceLevel>3</stork:QualityAuthenticationAssuranceLevel>
        <stork:spSector>DEMO-SP</stork:spSector>
        <stork:spInstitution>DEMO-SP</stork:spInstitution>
        <stork:spApplication>DEMO-SP</stork:spApplication>
        <stork:spCountry>LOCAL</stork:spCountry>
        <storkp:eIDSectorShare>true</storkp:eIDSectorShare>
        <storkp:eIDCrossSectorShare>true</storkp:eIDCrossSectorShare>
        <storkp:eIDCrossBorderShare>true</storkp:eIDCrossBorderShare>
        <storkp:RequestedAttributes>
            <stork:RequestedAttribute Name="http://www.stork.gov.eu/1.0/surname"
                                      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
                                      isRequired="false"
                                      />
            <stork:RequestedAttribute Name="http://www.stork.gov.eu/1.0/eIdentifier"
                                      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
                                      isRequired="true"
                                      />
        </storkp:RequestedAttributes>
        <storkp:AuthenticationAttributes>
            <storkp:VIDPAuthenticationAttributes>
                <storkp:CitizenCountryCode>LOCAL</storkp:CitizenCountryCode>
                <storkp:SPInformation>
                    <storkp:SPID>DEMO-SP</storkp:SPID>
                </storkp:SPInformation>
            </storkp:VIDPAuthenticationAttributes>
        </storkp:AuthenticationAttributes>
    </saml2p:Extensions>
</saml2p:AuthnRequest>
```


Listing C.2: A SAML Response message example. label


```
<saml2p:Response xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
```

```
            xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
            xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
            xmlns:stork="urn:eu:stork:names:tc:STORK:1.0:assertion"
            xmlns:storkp="urn:eu:stork:names:tc:STORK:1.0:protocol"
            xmlns:xs="http://www.w3.org/2001/XMLSchema"
            Consent="urn:oasis:names:tc:SAML:2.0:consent:obtained"
            Destination="https://localhost:3000/createdocument/samlreturn"
            ID="_22495689074129e9270c090120fb2288"
            InResponseTo="_de24b22e6e6d196c6b7b085608168f5c"
            IssueInstant="2015-05-08T17:36:00.441Z"
            Version="2.0"
            >
<saml2:Issuer
    Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">http://C-PEPS.gov.xx</saml2:Issuer>
<ds:Signature>
    <ds:SignedInfo>
        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
        <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" />
        <ds:Reference URI="#_22495689074129e9270c090120fb2288">
            <ds:Transforms>
                <ds:Transform
                    Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
                <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                    <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"
                                            PrefixList="xs"
                                            />
                </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
            <ds:DigestValue>FciRa70UeCuBOH0Xeos2gDveGSk=</ds:DigestValue>
        </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>lNcqa3y6QDNOmLuS9STFck81df3b36OQzDzbH2O+XHkYTaFoz4wMuG99pKFzQbaehpFMBkaOda+uS68ExO3Vq
    <ds:KeyInfo>
        <ds:X509Data>
            <ds:X509Certificate>MIIDIjCCAgqgAwIBAgIES6idsjANBgkqhkiG9w0BAQUFADBTMQswCQYDVQQGEwJFUzEOMAwGA1UE
CAwFU3BhaW4xDzANBgNVBAcMBk1hZHJpZDEOMAwGA1UECgwFSW5kcmExEzARBgNVBAMMCmxvY2Fs
LWRlbW8wHhcNMTMwODI4MTY0NzM1WhcNMTUwODI4MTY0NzM1WjBTMQswCQYDVQQGEwJFUzEOMAwG
A1UECAwFU3BhaW4xDzANBgNVBAcMBk1hZHJpZDEOMAwGA1UECgwFSW5kcmExEzARBgNVBAMMCmxv
Y2FsLWRlbW8wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCx+7zEQUbt8Ot9dByR5TuY
NPmTwaKhECtnkxYAx9tl+Ga3kFiv+dOSeDoQOhxkICG1ZMaG0Gtk0EQWjscC1cLwDXpAOdhOm5xh
ndFHxa3V3Y3zomFZA7A3nwP8wt17wDrc1xi2Ye8VrwO7Vjb/F8Rgutlcrvq+LF4g9nQLoRiKUq8w
NFvDlvhBbLr8hMVBMY/jY7tSQ85qLqKUkWRcoDttJjgWZoO1vpBS4m+FywtYdOoHnN6fv4Dvf1r2
rNLgebgBvSfwznxiulwW6FBLasYX0rYKP7RfwG8TJ+FumOgAjADj2LdwK25RZNg44XU2V1z1Fp37
fNXhfo08LpdD1ueFAgMBAAEwDQYJKoZIhvcNAQEFBQADggEBADMRGTWOmadZyCoJNluV+M7LopjC
9WpFUj76X0cAoNXmt12zYqYe8hjRmBr6l2V/FXT3/TyWET1nILG46zwwAxslw3b2LvkhU+9QcuB8
yC8TCJJ0kgsTZ/EUZPSbwUG7Nn2jkMiyjlyKcjSCQOzEKQyR1xYwyZG40+BPeGbYjmRgm6kcmCxY
USWoPwNyfke9gNT80f0pUj7wJ9YwWbTIz1rhf/h7rxoMYypXo+JXwaXW/Ra8v1uDcwfKpE5ZgwAU
nubLXF4A+H7/N7ZvB5XDwJ4W+99nuPsKfwacD8m1sflTXcEPzzhOq6iQ9anJT94/pMctnp827zXA
y66jvDrin5I=</ds:X509Certificate>
            </ds:X509Data>
```

```
    </ds:KeyInfo>
</ds:Signature>
<saml2p:Status>
    <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
    <saml2p:StatusMessage>urn:oasis:names:tc:SAML:2.0:status:Success</saml2p:StatusMessage>
</saml2p:Status>
<saml2:Assertion ID="_bdda25ac64d70c7b4505e29c05b9714a"
                IssueInstant="2015-05-08T17:36:00.441Z"
                Version="2.0"
                >
    <saml2:Issuer
        Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">http://C-PEPS.gov.xx</saml2:Issuer>
    <ds:Signature>
        <ds:SignedInfo>
            <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
            <ds:SignatureMethod
                Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" />
            <ds:Reference URI="#_bdda25ac64d70c7b4505e29c05b9714a">
                <ds:Transforms>
                    <ds:Transform
                        Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
                    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces
                            xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"
                                                PrefixList="xs"
                                                />
                    </ds:Transform>
                </ds:Transforms>
                <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
                <ds:DigestValue>a7QVGqvZfPHdawjfFnKpiEZdVzU=</ds:DigestValue>
            </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>
            ZFjtJirGuV5mX4GFeVi+1/rfBTrJTa6pOSjrfcv2KMhCq4GjWB7nykUYP9UBilHEURppizdFQ5sPhSiibs5a
            77bcbiZm5nIRkhdmzqBQLj/OwKC+4LR/b98iW7/ik1S2xjyMMqoNNbVucpz+ZeYvqMUcLy4IquKJ7bInR6t4
            8ZDt3vMJ+oW3Jfv4hY+tJtmsGVU4c3GgT7PnxRk97jvE4VFjV7ETohKqLcgYLMvnw5+Y72LUoh+sqzFxpc4P
            PXmCmX1tpvJZ3+C/zvKDaHLbdFq3k+5ixdIH+UHg8v2CozyJ+EJq1A1Sbk8liohAHqMxPInuONbd9NBHRDDa
            WcVwgQ==</ds:SignatureValue>
        <ds:KeyInfo>
            <ds:X509Data>
                <ds:X509Certificate>
                    MIIDIjCCAgqgAwIBAgIES6idsjANBgkqhkiG9w0BAQUFADBTMQswCQYDVQQGEwJFUzEOMAwGA1UE
                    CAwFU3BhaW4xDzANBgNVBAcMBk1hZHJpZDEOMAwGA1UECgwFSW5kcmExEzARBgNVBAMMCmxvY2Fs
                    LWRlbW8wHhcNMTMwODI4MTY0NzM1WhcNMTUwODI4MTY0NzM1WjBTMQswCQYDVQQGEwJFUzEOMAwG
                    A1UECAwFU3BhaW4xDzANBgNVBAcMBk1hZHJpZDEOMAwGA1UECgwFSW5kcmExEzARBgNVBAMMCmxv
                    Y2FsLWRlbW8wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCx+7zEQUbt8Ot9dByR5TuY
                    NPmTwaKhECtnkxYAx9tl+Ga3kFiv+dOSeDoQOhxkICG1ZMaG0GtkOEQWjscC1cLwDXpAOdhOm5xh
                    ndFHxa3V3Y3zomFZA7A3nwP8wt17wDrc1xi2Ye8VrwO7Vjb/F8Rgutlcrvq+LF4g9nQLoRiKUq8w
                    NFvDlvhBbLr8hMVBMY/jY7tSQ85qLqKUkWRcoDttJjgWZoO1vpBS4m+FywtYdOoHnN6fv4Dvf1r2
                    rNLgebgBvSfwznxiulwW6FBLasYX0rYKP7RfwG8TJ+FumOgAjADj2LdwK25RZNg44XU2V1z1Fp37
                    fNXhfo08LpdD1ueFAgMBAAEwDQYJKoZIhvcNAQEFBQADggEBADMRGTWOmadZyCoJNluV+M7LopjC
```

```
                9WpFUj76X0cAoNXmt12zYqYe8hjRmBr6l2V/FXT3/TyWET1nILG46zwwAxslw3b2LvkhU+9QcuB8
                yC8TCJJ0kgsTZ/EUZPSbwUG7Nn2jkMiyjlyKcjSCQOzEKQyR1xYwyZG40+BPeGbYjmRgm6kcmCxY
                USWoPwNyfke9gNT80f0pUj7wJ9YwWbTIz1rhf/h7rxoMYypXo+JXwaXW/Ra8v1uDcwfKpE5ZgwAU
                nubLXF4A+H7/N7ZvB5XDwJ4W+99nuPsKfwacD8m1sflTXcEPzzh0q6iQ9anJT94/pMctnp827zXA
                y66jvDrin5I=</ds:X509Certificate>
            </ds:X509Data>
        </ds:KeyInfo>
    </ds:Signature>
    <saml2:Subject>
        <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
                      NameQualifier="http://C-PEPS.gov.xx"
                      >urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</saml2:NameID>
        <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
            <saml2:SubjectConfirmationData Address="127.0.0.1"
                                           InResponseTo="_de24b22e6e6d196c6b7b085608168f5c"
                                           NotOnOrAfter="2015-05-08T17:41:00.441Z"
                                           Recipient="https://localhost:3000/createdocument/samlreturn"
                                           />
        </saml2:SubjectConfirmation>
    </saml2:Subject>
    <saml2:Conditions NotBefore="2015-05-08T17:36:00.442Z"
                      NotOnOrAfter="2015-05-08T17:41:00.441Z"
                      >
        <saml2:AudienceRestriction>
            <saml2:Audience>localhost</saml2:Audience>
        </saml2:AudienceRestriction>
        <saml2:OneTimeUse/>
    </saml2:Conditions>
    <saml2:AuthnStatement AuthnInstant="2015-05-08T17:36:00.442Z">
        <saml2:SubjectLocality Address="127.0.0.1" />
        <saml2:AuthnContext>
            <saml2:AuthnContextDecl/>
        </saml2:AuthnContext>
    </saml2:AuthnStatement>
    <saml2:AttributeStatement>
        <saml2:Attribute Name="http://www.stork.gov.eu/1.0/eIdentifier"
                         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
                         stork:AttributeStatus="Available"
                         >
            <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                  xsi:type="xs:anyType"
                                  >LOCAL/ES/12345</saml2:AttributeValue>
        </saml2:Attribute>
        <saml2:Attribute Name="http://www.stork.gov.eu/1.0/givenName"
                         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
                         stork:AttributeStatus="Available"
                         >
            <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                  xsi:type="xs:anyType"
                                  >Javier</saml2:AttributeValue>
        </saml2:Attribute>
```

```
        </saml2:AttributeStatement>
    </saml2:Assertion>
</saml2p:Response>
```

D

Annex 4

D.1 TADS Document Example

This Annex includes an example of a document generated with TADS. The digital signature is not visible, but on the digital version of the document the IST logo is depicted under the “Document Digital Signature” box, indicating the existence of the signature.

Trustworthy Validation Service

Instructions of Use

To validate this document please check the site http://tads.tecnico.ulisboa.pt and follow the instructions in the screen. You will be asked to accept the usage of the webcam in your computer to read the QRCode below. If you have no camera or are unwilling to use it for this purpose you will be asked to enter the number below the QR-Code.

Document Digital Signature

TADS-PT-158dcb10-f343-11e4-b5cf-2765b63f3dae

Certified Information

Retrieved From: ES-PEPS

On Date: Monday , 4th May 2015 1:16:37 CET

Disclaimer

This service is provided to you in accordance with the terms of the STORK 2.0 Memorandum of Understanding. Security and quality assurance measures which we deem to be appropriate have been implemented in accordance with the Memorandum. Please note however that the Memorandum excludes any assurances or liabilities for the availability, trustworthiness or accuracy of the identity information provided via its infrastructure. You should assess and decide for yourself whether these terms are acceptable for your individual use case.

Emails: tads@ama.pt, tads@tecnico.ulisboa.pt
Phone: +351 218 417 506
Address: Instituto Superior Técnico,
Av. Rovisco Pais, 1, 1049-001 Lisboa , Portugal

Attributes

NIF: 239173031

Nacionalidade: PRT

eIdentificador: PT-PT-14097180

NomeProprio: SIMON FILIPE

DataNascimento: 19900920

Idade: 24

NomeApelido: PACHECO ESPOSITO

Sexo: M

Emails: tads@ama.pt, tads@tecnico.ulisboa.pt
Phone: +351 218 417 506
Address: Instituto Superior Técnico,
Av. Rovisco Pais, 1, 1049-001 Lisboa , Portugal