FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Multi Factor Authentication of Users in U.Porto

André Malheiro

Master in Informatics and Computing Engineering

Supervisor: Prof. Dr. Pedro Brandão

Second Supervisor: Gil Silva

September 24, 2024

Multi Factor Authentication of Users in U.Porto

André Malheiro

Master in Informatics and Computing Engineering

Approved by the committee:

President: Prof. Dr. José Carlos Medeiros de Campos
Referee: Prof. Dr. João Marco Cardoso Silva
Supervisor: Prof. Dr. Pedro Miguel Alves Brandão

September 24, 2024

Abstract

Authentication security best practices are in constant evolution. The rise of cybercrime is associated with society's ever-growing digitalisation, forcing organisations to apply security measures to protect their digital infrastructures. For these reasons, in the last decade multi factor authentication (MFA) usage has seen major growth. Nowadays, while still not consensual, most organisations have some kind of MFA adoption.

Additionally, with the assistance of smartphones, there has also been a push to change the way we authenticate. While passwords still remain the dominant authentication factor, there exists a focus on developing secure and intuitive authentication methods to replace passwords.

Amidst this, the University of Porto's (U.Porto) authentication and authorisation infrastructure (AAI) is somewhat outdated. While using state-of-the-art technologies, including but not limited to OpenID Connect, Keycloak, and Shibboleth Identity Provider, it does not provide internal MFA flows to its users. Although users may authenticate through Autenticação.Gov (Auth.Gov), an external MFA flow, the AAI still requires an improvement regarding its authentication flows.

As such, this thesis proposes an AAI, compatible with U.Porto's requirements, to support MFA. The AAI is built around Shibboleth's Identity Provider as the central piece for authentication, integrating various authentication methods, including Time-Based One-Time Password (TOTP), X.509 certificates, traditional password authentication, and Fast-Identity Online (FIDO) authentication. A specialised MFA flow was developed to provide a seamless authentication experience, ensuring that users are authenticated using multiple secure factors. Additionally, the thesis describes the development, implementation and integration of a Flask application, Profile Management Application (PMA), that allows users to manage and configure their authentication factors.

Acknowledgments

First, I would like to thank my supervisors, Professor Pedro Brandão and Gil Silva, for all the support provided throughout this project, as well as for trusting me with it. Additionally, I would like to express my gratitude to UPdigital's team for providing me with a great experience during my time in their office.

I would also like to thank my parents for being who they are. Without your education, I would not have become the person I am today. I will forever be indebted to you.

To the rest of my family, especially Sofia and Maria Amélia, thank you. I couldn't have asked for a better support network.

To my friends, especially those who crossed my academic path, thank you for accompanying me on this journey. I won't forget the great memories we made these last few years.

The last thank you goes to my girlfriend, Filipa. Your unwavering support, love and motivation are what made me finish this work.

André Malheiro

Contents

List of Figures

List of Tables

Listings

Abbreviations

2FA	Two Factor Authentication
AAI	Authentication and Authorization Infrastructure
AD	Active Directory
AuthGov	Autenticação.Gov
CA	Certificate Authority
CRL	Certificate Revocation List
CPU	Central Processing Unit
CSR	Certificate Signing Request
FIDO	Fast Identity Online
HTTP	Hypertext Transport Protocol
HTTPS	Hypertext Transport Protocol Secure
IAM	Identity and Access Management
IdM	Identity Management
IdP	Identity Provider
IETF	Internet Engineering Task Force
JCEKS	Java Cryptography Extension Key Store
JWT	JSON Web Token
LDAP	Lightweight Directory Access Protocol
MFA	Multi Factor Authentication
NIST	National Institute of Standards and Technology
OAuth	Open-Authorization
OIDC	OpenID Connect
OS	Operating System
OTP	One Time Password
PKCRO	PublicKeyCredentialRequestOptions
PMA	Profile Management Application
PIN	Personal Identification Number
RAM	Random Access Memory
RP	Relying Party
SAML	Security Assertion Markup Language
SP	Service Provider
SSL	Secure Sockets Layer
SSO	Single Sign-On
TLS	Transport Layer Security
TOTP	Time-based One Time Password
TPM	Trusted Platform Module
U.Porto	University of Porto
UID	Unique Identifier
UML	Unified Modeling Language
URI	Uniform Resource Identifier
VM	Virtual Machine

Chapter 1

Introduction

In the last 20 years, the technology world has undergone significant transformations. First, there was the growth of the internet. Compared to the beginning of the millennium, the internet now reaches at least five times more users [3], connecting the whole world and most of its population in a few seconds. This has changed the way humans communicate, work and access information.

Following the internet revolution, the advent of smartphones marked a significant technological shift. Prior to 2007, the year the iPhone was first released, mobile phones were mainly used for calls, texting, and email. The introduction of smartphones transformed this landscape, with mobile devices overtaking computers as a primary means of accessing the internet, a trend projected to continue to grow in the future [4].

These advancements, conjoined with the proliferation of applications and websites, contributed to a substantial growth in the number of existing accounts, as these platforms require user registration, resulting in individuals having to manage multiple accounts. In order to avoid these issues, technologies such as single sign-on (SSO) and password managers were created to assist users. Despite this, existing apps are still highly dependent on passwords as their main authentication factor, which may be a security liability for the services.

Considering this, researchers and companies have undertaken efforts, such as the push for multi factor authentication adoption, to mitigate these risks and simplify life for their users. Managing and protecting these digital identities is a challenge and will be one of the major focal points of this dissertation.

1.1 Context

The following subsections present the background knowledge required to understand the project and the work developed. Additionally, it also introduces the partner institution for this thesis.

1.1.1 Authentication and authorisation

Authentication and authorisation are two concepts that go hand in hand, wrongfully and frequently grouped together as one joint idea. Despite the key differences, both ideas are complementary,

composing a larger concept known as the triple a's - authentication, authorisation and accounting.

Rountree (2013)[5] defines authentication as "the process of verifying that you are who you say you are" (13). For example, the authentication process of a system restricting access to certain resources can be divided into two steps - first, the user provides identity information followed by proof of that identification. This is performed using authentication factors, which can be divided into five separate categories:

- Something you know - a memorised secret, e.g. a password, PIN;
- Something you have - e.g. a smart card, token;
- Something you are - e.g. biometrics features, such as voice, face or fingerprint;
- Somewhere you are - e.g. granting access if a user is within a specific geographical area, IP address verification;
- Something you do - newer and less common factor, e.g. keystroke dynamics, mouse movement patterns.

On the other hand, authorisation consists of verifying what a user is allowed to do with a certain resource, i.e. his permissions. Although not mandatory, this process commonly occurs after the authentication process has ended. Therefore, systems must have policies in place which define users' permissions. For instance, online document editor services, such as Google Docs, allow its users to define authorisation policies for each of their documents, specifying who may access it and the allowed operations (read, write or comment) on that document.

1.1.2 Multi-factor Authentication

Since its introduction over 60 years ago, passwords have been the dominant authentication factor. In fact, it was not until the 1980s where other authentication factors, such as One-Time Passwords (OTP) or Public Key Infrastructure became available to consumers and enterprises, which naturally led to the birth of Two-Factor Authentication (2FA). This concept works on the basis that a user must configure two different categorised authentication factors in order to prove its identity, i.e. to authenticate to a system. On the other hand, MFA extends 2FA, requiring two or more authentication factors in order to complete an authentication successfully.

After its conception, 2FA was mainly implemented in some larger enterprises and governmental agencies. However, during the 2000s, the idea gained more traction, largely motivated by the rise of data breaches and cyber-attacks. This new perspective, where everyone could be a victim of the next attack, bolstered the need for a strengthened and safer Internet.

At first, following NIST's 2004 guidelines [6], entities decided to address the challenge by implementing password policies, such as password composition requirements (e.g. mandatory use of non-alphabetic characters) and periodic password changes, in an effort to strengthen users passwords. These, however, were proven ineffective by researchers in the following years [7] [8].

As a result, in 2010, Google released Google Authenticator, their 2FA OTP system [9], a feature at first available only to their domain customers but quickly released to all Google accounts in 2011.

In the following years, other tech giants, such as Microsoft, Apple, and X (previously Twitter), also released features enabling 2FA authentication on their platforms. Since then, 2FA and MFA adoption has grown, especially aided by enforcement by the biggest companies and systems in recent years.

Currently, MFA is a requirement by many platforms. Even though it does not prevent data breaches by itself, it is a second barrier to security, proven by security reports, such as Google's in 2023 stating that 86% of data breaches involved stolen credentials [10].

1.1.3 Federated Identity

To address the problem of users having several accounts and passwords, authentication methods, such as Single Sign-On (SSO), were created. SSO aims to centralise authentication and allow users to log in once and access multiple applications or systems without providing several credentials for each one. Common examples are Google's [11], Facebook's [12] and Apple's [13].

Federated identity takes it a step further - it is a way of distinct systems to get access to a user's identity information. Take the case of a university student researching for a project. Its university does not have direct access to some research databases. However, it can develop a trusting relationship with a federation with access to these databases. So, whenever the student tries to access those services, the federation accesses the student's information provided by its university and allows the user access to the desired resources. The identity data is never saved in the federation, remaining only in the university information system.

Federated identity requires the existence of an identity provider (IdP), which is the entity that holds identity information. It also needs the existence of a service provider (SP), which is the entity that provides services to other entities and relies on IdP to provide identity information.

This model has several benefits, such as:

- Credentials are secure from the SP. Only the IdP has access to the user credentials;
- No negative impact on a user experience;
- Applications are only required to deal with the authorisation part. Authentication is not its responsibility;
- Reduces the amount of account management and the number of usernames and passwords.

On the other hand, it also contains some disadvantages:

- If the credentials or the IdP is compromised, a user's application would also be compromised;
- Requires specialised infrastructure;
- Federations are required to conform to the same protocols/standards. Otherwise, it would be impossible for communication to work.

1.1.4 Host institution

This thesis was developed in collaboration with UPdigital, a U.Porto division responsible for designing and maintaining its IT services.

As the second biggest university in Portugal, U.Porto is composed of 14 faculties, 48 research units, more than 40.000 active members, including students and researchers, and 115.000 former students. Similarly to other universities, U.Porto also adheres to the federated identity model, meaning it must possess specialised authentication and authorisation infrastructure (AAI).

As a university, U.Porto provides internal and external services to its members, whose accesses are controlled. While some services are restricted to personnel, other services, such as e-mail accounts, are supplied to all its members. In addition to this, access to some services must be configurable, as throughout the academic year, students require services such as laboratories.

Currently, U.Porto's AAI comprises a centralised database, directory services and authorisation services. Regarding authentication, users have a choice of authenticating using their usernames and passwords, or by using Portugal's state SSO application, Autenticação.Gov [14].

1.2 Problem definition

As stated in the previous section, just as information security knowledge and maturity develop, so does the area of cybercrime. Data breaches occur every year, leading to the leak of billions of passwords, originating considerable costs for enterprises and governments [15]. For example, recently occurred the new biggest credential data breach, the RockYou2024 breach, which reportedly exposed almost 10 billion passwords at once [16]. This breach, and many others, are among the reasons why businesses maintain a push for MFA adoption, among other defences and practices [17].

Even though universities may not be considered top targets for data breaches, the UK government released in 2023 a survey in which it reveals its universities were more likely targeted than average businesses. Further, the report also states that, compared to the previous year, businesses saw a decrease in the number of attacks and breaches, while the numbers reported for education institutions remained consistent [18].

While no data breach has yet been reported by U.Porto, the username and password authentication flow provided to users is insecure, considering the amount of sensitive data an attacker could reach just by knowing the username and password combination. While Autenticação.Gov offers MFA, and its use is steadily growing, it is still not used by every Portuguese citizen [19].

As a result, U.Porto faces a need to renovate its authentication system. While it must maintain the federated identity model, it must add and update its authentication flows by implementing new authentication factors and MFA in order to improve the authentication security, further protecting the U.Porto from data breaches and security incidents.

1.3 Goals

The primary goal of this thesis is to study the use of MFA in U.Porto's identity management solutions. Although there is no lack of research on the use of MFA in AAIs, this research can help contribute to a niche subset, such as the federated context. As such, this dissertation is expected to contribute to the federated AAI discussion.

On the other hand, there is also a focus on the practical aspect of the work. Having studied the U.Porto's AAI and state-of-the-art solutions, there is an expectation to deliver a tested prototype of an AAI for U.Porto. This new system must comply with current security best practices, such as MFA, and also maintain compatibility with the U.Porto distinct requirements.

1.4 Structure

This document is divided into five other chapters. Chapter 2 explores identity and access management solutions while presenting authentication and authorisation frameworks and protocols. Additionally, it explores modern authentication factors while outlining its security recommendations and implementation guidelines.

Chapter 3 provides an overview of U.Porto's current AAI, identifying its limitations and flaws regarding present-day standards.

Chapter 4 identifies U.Porto's requirements and outlines the proposed solution and its architecture.

Chapter 5 described the implementation of the solution and the digital deployment environment.

Lastly, Chapter 6 briefly summarises the developed work and identifies future work possibilities.

Chapter 2

Related Work

This chapter will present and analyse identity management (IdM) tools, protocols, and security measures in authentication.

First, there is a focus on a subset of current protocols used in IdM. Then, the Lightweight Directory Access Protocol (LDAP) is presented as a directory management protocol. Next, emerging passwordless technology, Fast-Identity Online (FIDO) is introduces. Then, the Autenticação.Gov (Auth.Gov) authentication solution and the X.509 technology are presented. Afterwards, the focus shifts to open-source IdM solutions which implement the protocols above. To close the chapter, the focal point changes to the security of several authentication factors and current implementation guidelines.

2.1 Protocols

In the following subsections, the state-of-the-art authentication protocols are presented. Additionally, the LDAP protocol is also introduced.

2.1.1 SAML 2.0

The Security Assertion Markup Language (SAML) is an Extensible Markup Language (XML) based standard for describing and exchanging security information between entities. The latest version, SAML 2.0[20], was ratified as a standard in March 2005, replacing SAML 1.1. This new version, the only major update SAML received, added several new features, causing it to be incompatible with previous protocol versions.

Even though this protocol version was released more than 18 years ago, it is still widely used worldwide, predominantly regarding federated identity, which is often the case for universities. The adoption throughout academia made several other entities providing services to universities use this standard.

In past years, however, other protocols such as OAuth 2.0 and OpenID Connect 1.0, both of which are presented in the following subsections, have taken a larger share of the market. The last decade's technology trends can explain this popularity. By the time SAML 2.0 was released, two

years before the first iPhone was released, there were no mobile applications. So, naturally, SAML was not designed for mobile applications, only web applications. Even though mobile applications are progressively relevant, SAML has remained one of the most used protocols, a feat expected to be maintained, in large part due to its mandatory usage by large federations. An example is EduGain [21], a federation that provides services to more than 27 million students and researchers worldwide.

SAML assumes three key roles in its specification [20]:

- Asserting party, often named Identity Provider (IdP), which is the one who makes SAML assertions;
- Relying party, often named Service Provider (SP), is an entity that uses assertions it has received. To rely on information from the IdP, there must previously exist a trust relationship between the two parties;
- Subject, often named User, is an entity that can be authenticated. It can be a human or some other kind of entity.

Additionally, SAML defines four other basic concepts[20]:

- **Assertions** define the syntax and semantics to describe authentication, attribute, and authorisation information;
- **Protocol** defines the syntax and semantics to describe protocol messages to carry information between systems.
- **Bindings** define which messaging protocols (e.g. HTTP or SOAP) are used to transport SAML messages;
- **Profiles** are combinations of assertions, protocols, and bindings to support a defined use case.

SAML also implements SSO. Figure 2.1 illustrates a simple SAML login flow:

1. The User tries to access a resource from the SP;
2. The SP verifies the User has no valid session on its domain. It then creates a SAML request and redirects the User to its IdP;
3. The IdP verifies if the User has a valid session on its domain. If it has, then it skips to step 5. Otherwise, it provides the User with an authentication challenge;
4. The User logs in by correctly answering the previous challenge, and a security context for the User is created on the IdP;
5. The IdP generates a SAML response and returns it to the User;
6. The User is redirected to the SP and delivers the SAML response to the SP, which grants the User access to the resource.

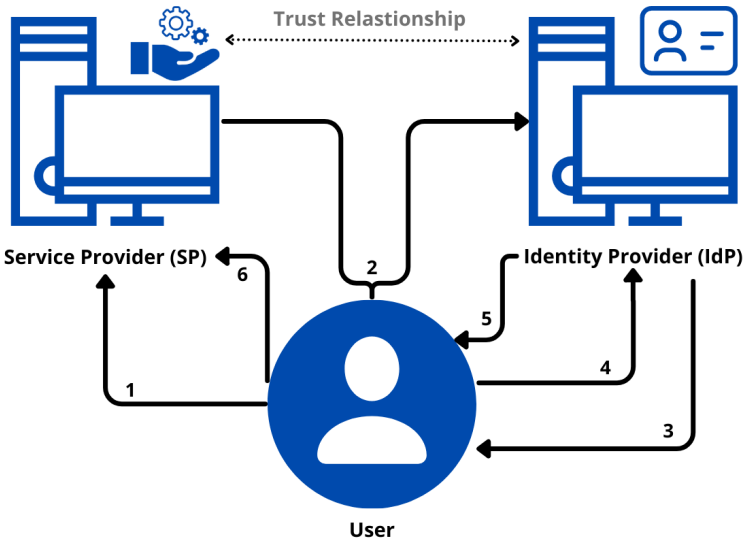

Figure 2.1: SAML example authentication flow

2.1.2 OAuth 2.0

OAuth is an authorisation framework created in 2007 to address issues in third-party authentication. At that time, if someone wanted to allow a third-party application, T, to access information from another application, A, they would need to provide A credentials to T. Even though it used to be expected, nowadays, it is considered to be a significant security risk and design flaw. OAuth sought to mitigate and fix this issue. Instead of requesting a user's credentials, the external application redirects the user to an OAuth server, which authorises and supplies the user with a token specifying its lifetime, scope, and access attributes, among others.

The technology landscape has completely changed since the current version, released in 2012. Nowadays, mobile phones have become the main access point to the internet, replacing computers. In addition, internet users have more than doubled, from 2,387 million in 2012 to 5,300 in 2022 [3]. So, to keep up with this ever-changing panorama, contrary to SAML, there has been a continuous effort to patch and improve OAuth. These patches have led to a current effort to consolidate OAuth 2.0 and its descendants in a new version, OAuth 2.1 [22]. Its specification is currently under discussion, meaning it is not yet ready for use. However, this draft presents some practices that can benefit existing systems.

Figure 2.2 demonstrates the OAuth 2.0 abstract flow. OAuth also defines different flows based on the abstract flow, which can be found in its specification [1].

Even though steps A and B do not involve the Authorisation Server (AS), the preferred method for obtaining an authorisation grant from the Resource Owner (RO) is using the AS as an interme-

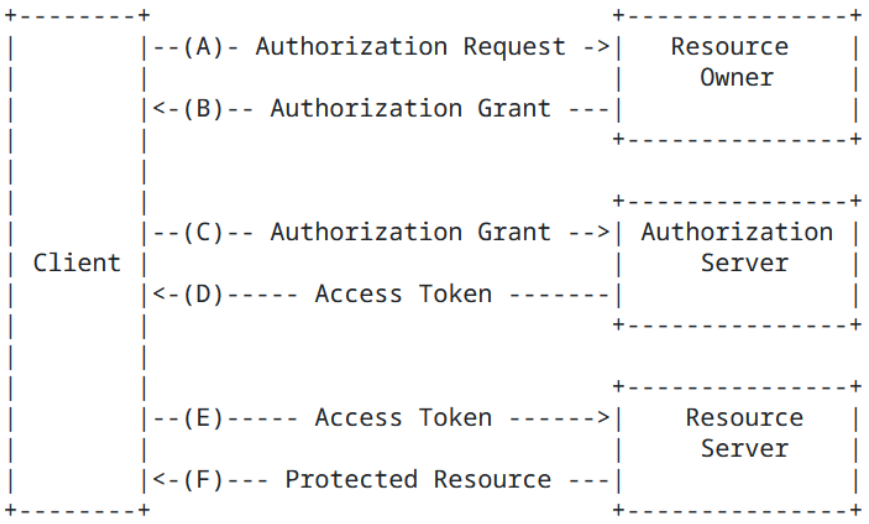

Figure 2.2: OAuth 2.1 abstract flow [1].

diary during these steps. The client uses the access token to gain access to the desired resource. This token is either a string or a JSON Web Token (JWT), a compact way to share information between parties securely.

There are also some fundamental rules regarding access tokens. They must not be read or interpreted by the client, mustn't convey information about the Client, and should only be used to make requests to the Resource Server.

In its specification [1], OAuth 2.0 assumes four key roles:

- Resource Owner - An entity capable of granting access to a resource;
- Resource Server - The server which hosts the protected resources. Capable of responding to resource requests through the use of access tokens;
- Authorisation Server - The server can issue access tokens to the client after authenticating the resource owner and obtaining authorisation. It can be the same server as the resource server;
- Client - Application making protected resource requests.

2.1.3 OpenID Connect 1.0

OpenID Connect 1.0 (OIDC) [2] is an identity authentication protocol implemented on top of OAuth 2.0. It was released in 2014 by the OpenID Foundation and is the third generation of OpenID technology, succeeding the obsoleted OpenID 1.0 and OpenID 2.0. It originated from a need to standardise authentication on top of OAuth.

Like OAuth 2.0, it was designed to support web and mobile applications. As of this writing, OIDC is still actively developed by several working groups. The draft specifications focus on areas and industries such as mobile network operators, security, privacy, and government.

OIDC shares similarities to OAuth and SAML. It defines an IdP as an OpenID Provider (OP) and the RP as an OpenID Client or Client. It supports different authentication flows, which begin with an authentication request from a Client to the OP, specifying the flow requested. OPs do not need to keep all existing flows. Figure 2.3 (OpenID Foundation 2014) illustrates an OIDC abstract flow.

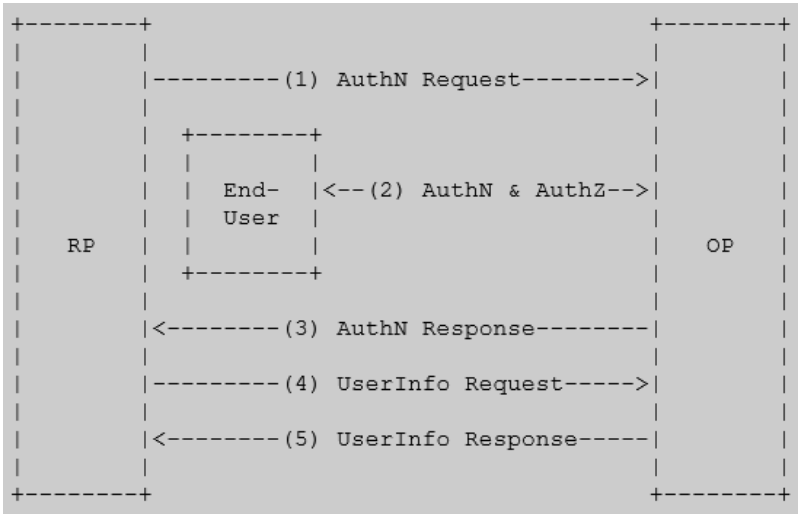

Figure 2.3: OpenID Connect 1.0 abstract flow [2].

The primary extension to OAuth is an ID token, represented as a JWT, and contains information regarding the outcome of the authentication request and the token itself. Tokens must be signed and may also be encrypted, using JSON Web Signatures (JWS) and JSON Web Encryption (JWE), guaranteeing authentication, integrity, and non-repudiation [23].

2.1.4 LDAP

The Lightweight Directory Access Protocol [24] is used for accessing directory information and data. It provides a standardised way to interact with directory services, which store and organise information about users, systems, resources, and other network entities.

Its current version, LDAPv3, was released in late 1997, making it one of the oldest directory management protocols. Even though it launched more than 25 years ago, work is still being done to improve the standard, with revisions being released occasionally. This is partly due to its high importance in large enterprises. This extensive use can be explained by LDAP's track record and its stability throughout the years.

LDAP is compatible with several applications, serving as the underlying protocol for many directory services. Besides, its lightweight features give it an edge compared to modern protocols, using an efficient and compact binary format for encoding and decoding. It also reduces costs due

to its long, persistent connections for communication with a directory server. As newer HTTP-based protocols use short-lived connections, establishing new connections is costly, especially when requiring TLS sessions.

In addition, LDAP takes security seriously. It can secure communication by enabling LDAP over SSL (LDAPS). Sensitive information such as passwords and user credentials is encrypted. Additionally, LDAP offers Access Control Lists (ACLs) to manage permissions and restrict access to directory entries and attributes based on user roles or groups. Finally, LDAP uses hashing algorithms to store passwords securely, ensuring that user credentials are not stored in plaintext. LDAP also includes password policy functionalities and the possibility of using two-factor authentication mechanisms.

2.2 FIDO

The FIDO Alliance [25] is an industry consortium formed in 2012 to address online security concerns, explicitly password-based authentication. Originally comprised of several corporations such as PayPal, Lenovo, and Infineon, it can nowadays count on the support of collaborators from industry leaders such as Google, Microsoft, Apple, Samsung, etc. and many governmental institutions like NIST.

The alliance aimed to provide stronger authentication through public key cryptography techniques. Since its inception, the conglomerate has released three specifications [26]:

- Universal 2nd Factor(U2F), now known as Client to Authenticator Protocol 1 (CTAP1) [27];
- Universal Authentication Framework (UAF) [28];
- Client to Authenticator Protocol 2 (CTAP2) [29].

The first two specifications, UAF and U2F, composed the FIDO1 project, which focused on replacing traditional passwords with biometrics and hardware tokens. However, these components lacked standardisation, leading to a lack of adoption of the technologies. Eventually, FIDO2 (CTAP2 + WebAuthn [30]) was released, standardising the approach to passwordless authentication. As a result, several browsers and operating systems started implementing the standard, allowing users to take advantage of these technologies.

There are two parties in every FIDO2 operation: the user, who wants to access some protected resource, and the RP, who owns the protected resource.

Besides, a key component of the project is an authenticator. This authenticator, owned by the user, must generate public-private key pairs to verify the user's identity during authentication. They also must support an authorisation gesture to verify user presence during authentication. Authenticators can be separated into two categories:

- Platform authenticator - typically integrated into the device (laptop or smartphone). Devices must contain a Trusted Platform Module (TPM) [31], where the authentication data

will be stored. Examples of this type of authenticator are Apple's TouchID and FaceID or Microsoft's Windows Hello.

- Roaming authenticator - also known as cross-platform or external authenticators. It may be used across several client devices. It may connect to the client using USB, Lighthing Bolt, Bluetooth, or NFC. In the future, smartphones may also be used as roaming authenticators. Examples of cross-platform authenticators are hardware security keys, such as Yubico's YubiKeys.

The specifications rely on two different operations: registration and authentication. When registering with an online service, the user creates a unique key pair for that service, retains the private key, and shares the public key with the server, which is then stored for later use. The authentication is completed by users proving the possession of the private key to the online service, accomplished by a challenge-response type of authentication. The private keys are only available after the client locally unlocks them via user-friendly actions such as entering a PIN, providing a fingerprint, etc.

WebAuthn also supports, if desired by the RP, the use of attestations to verify the authenticity of an authenticator and provide assurances regarding its features. This is achieved through an attestation statement, which is a signed piece of data generated by an authenticator that proves the authenticity and integrity of the device and key pair used during registration. It provides information about the authenticator's provenance and can be used by the relying party to assess the trustworthiness of the device. The goal of these elements is to assist RPs in filtering trustworthy devices from unknown devices. Attestation statements can be separated into five types:

- Basic attestation (Basic) - the attestation key pair is specific to an authenticator model, and authenticators of the same model may share the same key pair.
- Self attestation (Self) - the authenticator does not have an attestation key pair. Instead, it uses the key pair generated for registration.
- Attestation CA (AttCa) - based on a TPM, the authenticator can generate multiple attestation key pairs and request an Attestation Certificate Authority to issue an attestation certificate, which can be shared with the RP.
- Anonymization CA (AnonCa)- the authenticator requests the Attestation CA to generate the attestation key-pair certificates. These certificates do not provide uniquely identifiable information.
- No attestation statement (None) - no attestation information is available.

WebAuthn also defines the structure and method by which the authenticator provides cryptographic proof of its authenticity during the registration process. This is referred to as an attestation statement format, which specifies the data fields, such as the attestation type, cryptographic algorithms, and any additional metadata, along with the method used to sign the attestation statement,

ensuring the integrity and origin of the key pair generated by the authenticator. Table 2.1 outlines the different attestation formats supported.

Table 2.1: Attestation statement formats overview.

Format name	Description	Supported attestation types
Packed	Optimised attestation format for WebAuthn. Uses a very compact but extensible encoding method	Basic, Self and AttCA
TPM	Generally used by authenticators that use a TPM as their cryptographic engine	AttCA
Android Key	Used by devices operating on Android 7 or later. The attestation statement is produced in a secure operating environment, but the data is produced outside this environment	Basic
Android SafetyNet[1]	The authenticator data is completely controlled by the caller of the SafetyNet API (typically an application), and the attestation provides statements about the health of the device/platform and the identity of the application	Basic
FIDO U2F	Used with FIDO U2F authenticators	Basic, AttCA
None	Used to replace any authenticator-provided attestation statement if the RP indicates no attestation information is needed	None
Apple	Exclusively used by Apple for their devices supporting WebAuthn	AnonCA

WebAuthn also specifies two distinct types of credentials - Discoverable Credentials and Non-Discoverable Credentials, also known as Resident Keys and Non-Resident Keys, respectively. Discoverable credentials are securely stored in persistent memory on the authenticator itself and linked to the user account on the RP. When registering to an RP, a public-private key pair is generated by the authentication, which stores the private key and shares the public key with the RP. On registration, the authenticator also stores a credential ID and the RP ID to associate the generated key with the user handle (e.g. e-mail, username, phone number). In contrast, non-resident key registration also generates a public-private key pair and shares the public key. However, the authenticator only stores the metadata related to the registration (e.g. user handle) in its memory. Whenever the authenticator requires access to a specific private key, the private key is derived from the authenticator master key and other specific registration data, such as RP ID. This may seem like a small difference, but it changes the login process depending on the type of credentials used.

[1]This API has been deprecated by Google since January 2024. https://developer.android.com/privacy-and-security/safetynet/deprecation-timeline

Figures 2.4 and 2.5 illustrate, respectively, the discoverable credential authentication process and the non-resident key authentication process.

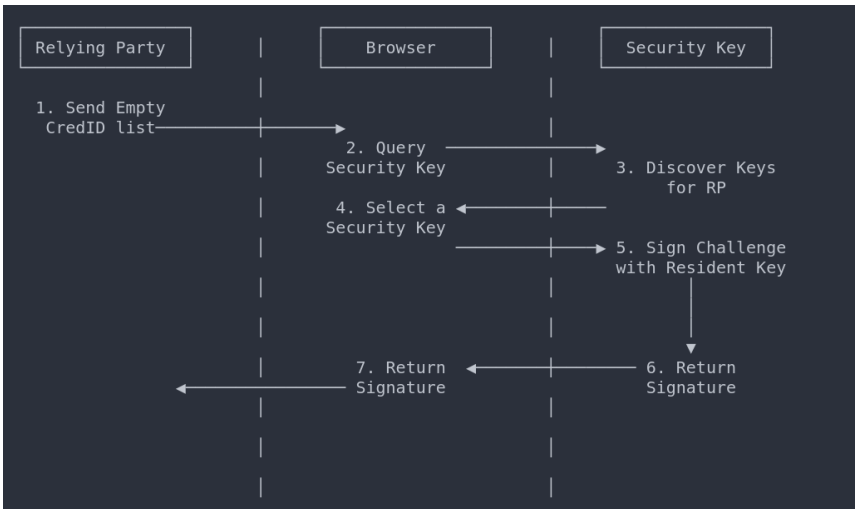

Figure 2.4: Discoverable credential authentication process.[2]

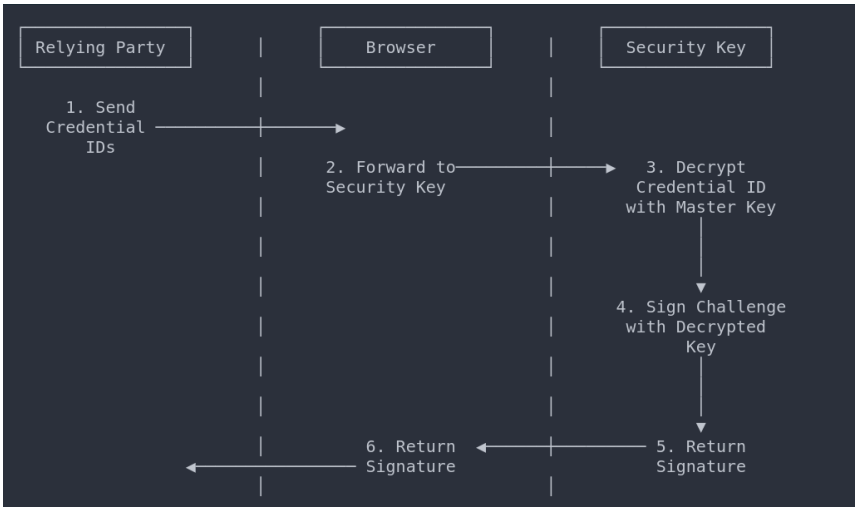

Figure 2.5: Non-discoverable credential authentication process.[2]

2.3 Autenticação.GOV

Autenticação.Gov (Auth.Gov) is a Portuguese digital identity system that enables secure authentication for citizens accessing various public and private online services. It integrates multiple authentication mechanisms, including the Citizen Card and Mobile Digital Key, allowing users to verify their identity. The system is designed to ensure high levels of security and user convenience while complying with European and national regulations for electronic identification.

[2]William Brown's figure, 2023, "Firstyear's blog-a-log",https://fy.blackhats.net.au/blog/2023-02-02-how-hype-will-turn-your-security-key-into-junk/

The Citizen Card operates via physical smart cards that store cryptographic keys and personal data, enabling two-factor authentication (2FA) through a combination of the card and a PIN code. Mobile digital key is an alternative mechanism that allows users to authenticate using a combination of the mobile digital key PIN and either a one-time password (OTP) or by reading a QR code using the mobile application. The OTP can be received by SMS, e-mail, or mobile application. Both methods rely on strong cryptographic protocols to safeguard personal information and ensure the integrity of online transactions. As both methods always require two different authentication factors, Auth.Gov can be considered an MFA flow.

2.4 X.509

X.509 authentication is a cornerstone of modern digital security, providing a robust framework for secure network communications. Originally created to address the need for secure identification and communication in increasingly interconnected digital environments, it was developed as part of the X.500 series of standards for directory services. X.509 has become the most widely adopted standard for public key infrastructure (PKI) and digital certificates.

The primary purpose of X.509 authentication is to provide a standardised method for authenticating entities using digital certificates, thereby ensuring the confidentiality, integrity, and authenticity of communications. This is achieved through the use of cryptographic key pairs and a hierarchical system of trusted certificate authorities (CAs).

An X.509 certificate is a digital document that binds a public key to the identity of its owner, typically a person, organisation, or device. The structure of an X.509 certificate typically includes the following components:

- Version: Specifies the version of the X.509 standard that the certificate adheres to. The most common versions are v1, v2, and v3, with v3 being the most widely used due to its support for certificate extensions;
- Serial Number: A unique identifier assigned by the issuing certificate authority (CA) to distinguish the certificate from others;
- Signature Algorithm: The algorithm used by the CA to sign the certificate. This includes both the hash function and the encryption algorithm, such as SHA-256 with RSA;
- Issuer: The entity that issued the certificate, usually a CA. This field contains the name of the CA in a specific format (Distinguished Name, DN);
- Validity Period: The period of time during which the certificate is valid, defined by the "not before" and "not after" dates;
- Subject: The entity to whom the certificate belongs. This can be an individual, an organisation or a device identified by its DN;

- Public Key: The public key associated with the subject, which is used in conjunction with the corresponding private key for encryption and decryption operations;
- Extensions: X.509 v3 certificates can include various extensions to add additional information, such as key usage restrictions, alternate names, or policy information;
- Signature: The digital signature of the CA, which confirms the authenticity of the certificate's contents. This signature is created using the CA's private key.

X.509 authentication relies on the use of public and private key pairs, where the private key is kept secret by the owner, and the public key is distributed via the X.509 certificate. The authentication process generally follows these steps:

1. Certificate Issuance: The subject generates a key pair and submits a Certificate Signing Request (CSR) to a CA. The CA verifies the identity of the subject and issues an X.509 certificate, signing it with its private key;
2. Certificate Verification: When a subject presents an X.509 certificate to a relying party (e.g., during a TLS handshake), the relying party verifies the certificate's authenticity. This involves checking the CA's signature, ensuring that the certificate has not expired, and confirming that it has not been revoked;
3. Trust Chain Validation: X.509 certificates are often part of a certificate chain, where an end-entity certificate is signed by an intermediate CA, which in turn is signed by a root CA. The relying party must validate each certificate in the chain up to a trusted root CA, which is pre-installed in the system's certificate store;
4. Authentication: Once the certificate is verified, the relying party can trust that the public key in the certificate belongs to the subject. The subject can then use its private key to perform cryptographic operations, such as signing data or establishing a secure session.

However, the security of X.509 systems hinges on the integrity of the issuing CAs and the proper management of private keys. Compromised CAs or poorly managed private keys can lead to significant vulnerabilities, such as man-in-the-middle attacks or unauthorised access. To mitigate these risks, mechanisms like certificate revocation, strict key management practices, and regular audits are essential. As digital threats evolve, maintaining the robustness of X.509 authentication is critical to securing global digital communications.

2.5 IdM Solutions

2.5.1 Shibboleth

The Shibboleth [32] software is an open-source web-based single sign-on system (SSO). Initially, the project stemmed from an Internet2 [33] Middleware Initiative in 2000. Since 2013, the Shibboleth Consortium, comprising entities such as universities and federations, has managed the project.

Its goal was to connect organisations with incompatible AAIs that support FIdM. Implementing SAML from the start, the Shibboleth project implemented features to make up for the shortcomings of early SAML versions. As a result, some of the components ended up contributing to the evolution of SAML, specifically to the development of SAML 2.0.

Shibboleth has four available products: an IdP, an SP, an Embedded Discovery Service, and a Metadata Aggregator. Each organisation can integrate or deploy these components separately, depending on its needs. This dissertation focuses on the IdP, mainly used in higher education organisations, where large federations and universities work together and exchange identity information through SAML.

Even though Shibboleth has been a household name in the last decades, it is still heavily maintained and under continuous development [34]. In addition, it is actively extending an official plugin to its IdP to support the use of the OIDC framework, meaning Shibboleth IdP supports OIDC. However, this plugin is only available from version 4.1 onward.

Using Shibboleth's IdP has several benefits and some limitations. On the one hand, it is a free product that is continuously improved, used by thousands of academic entities/federations, compatible with Windows and Linux, customisable, and several plugins to extend the tool. On the other hand, its configuration is based on XML files, it supports limited protocols, and the software reaches its end-of-life relatively fast.

2.5.2 KeyCloak

KeyCloak [35] is an open-source IAM tool. It is a project owned by Red Hat, and, in a similar fashion to Shibboleth, there is active development to improve the solution.

It provides several features, such as:

- SSO;
- Compatibility with SAML 2.0, OAuth 2.0, and OIDC;
- Default support and synchronisation of LDAP and Active Directory;
- Web-based GUI console for administration and configuration;
- Built-in support for social networks such as Google, Facebook, GitHub, etc;
- Support for MFA;
- Extensively customisable;
- Authorisation services. In addition, the administrator may define and manage the policy and the permissions.

However, exporting its configurations to another server is complex (e.g., when exporting configurations from development to production).

2.6 Security in authentication

Due to its importance, security in the authentication process has been a significant topic for discussion. Companies are constantly looking to fortify their authentication system - the barrier separating their private resources from the outside world.

NIST (National Institute of Standards and Technology), one of the most renowned entities regarding cybersecurity, periodically releases guidelines spanning several topics. As the dissertation focuses on authentication and federated identity management, the policies followed were respectively [36] and [37]. Both are draft guidelines, as NIST considered the complete approaches in 2017 to require an update.

Considering this, the following subsections present several authentication factors and requirements the server should or must implement to secure its authentication process.

2.6.1 Memorised Secret

Memorised secrets, a type of "Something you know" factor, are secret values intended to be remembered by the user, commonly known as passwords or PINs (when composed of numeric characters).

Although some reports have called for the replacement or discontinuance of passwords altogether, reports still show passwords remain, by a wide margin, the preferred [38] authentication method by users. Due to its importance and susceptibility, organisations such as NIST continue releasing guidelines to improve the security of password usage. The following list contains some of the guidelines from the latest release of NIST guidelines:

- Password length must be between 8 and 64 characters when chosen by the user;
- Password length must be at least six characters when chosen by the server;
- Hints must not be accessible to an unauthenticated user;
- When processing requests to establish or change memorised secrets, the new secret should be compared to lists of commonly used, expected or compromised values;
- Guidance should be offered to the user when choosing a new password, such as a password-strength meter, especially after rejection of a memorised secret;
- Limit the number of consecutive failed authentications;
- No composition rules should be imposed, such as requiring mixtures of different character types;
- Periodic change of the secret should not be required. Changes should be made if there is evidence the secret is compromised;
- Use of paste functionality should be permitted, facilitating the use of a password manager (recommended by NIST);

- Users should be allowed to display the secret;
- Characters may be displayed shortly after entry to verify if it is correct. Especially applicable to mobiles.

Additionally, NIST provides recommendations regarding the storage of memorized secrets, though they were not deemed relevant enough to be included in this document. However, readers who wish to deepen their knowledge may do so by reading [36], pages 16-17.

2.6.2 Look-Up Secret

A look-up secret is categorised as a "Something you have" factor of an authenticator. It can be described as a physical or electronic record storing a set of secrets shared between the client and the server. It is commonly used as a recovery key when users lose access to their selected second factor. Another example of this type of factor is grid cards, which are still used by some banks in Portugal.

Regarding recovery codes, NIST recommendations limit successful use to once. Similarly, NIST advocates restricting the use of each cell of a grid card to once.

2.6.3 Out-of-Band Devices

These physical devices communicate with the server over a secure (secondary channel), categorised as a "Something you have" factor. These appliances use a temporary secret generated by the server, which aims to prove the user's possession of the device. Therefore, channels that do not verify the control of the device are not accepted, such as email services or VOIP endpoints.

There are three distinct ways it can operate:

- User transfers a secret received by the out-of-band device via the secondary channel to the secret via the primary channel. Examples are SMS tokens or applications receiving a code from the server;
- User transfers a secret received via the primary channel to the out-of-band device, which will transmit to the server via the secondary channel. Examples are using QR codes or typing the code into an app on mobile devices;
- The user compares secrets received in the primary and secondary channels and must approve the authentication on the secondary channel. An example is push notifications. This method has been removed from the December 2022 draft.

NIST also presents some requirements:

- If the authenticator is a secure application on a mobile, the server may send a push notification to the device;
- Authentication should be considered invalid if not completed within 10 minutes;

- Servers should accept a secret only once during the validity period.

This type of factor can be considered multi-factor when it requires the verification of an additional factor, either by using a memorised secret or a biometric characteristic before the user completes the authentication. Each use of the out-of-band device should require the presentation of the activation factor, which must differ from the operation of unlocking the host device (e.g. smartphone).

SMS-based authentication has a few advantages regarding other types of factors:

- Ease of adoptions for users;
- SMS is a standard feature in mobile plans all over the world;
- Users do not need smartphones. A simple mobile is enough.

However, there are also costs associated with SMS-based authentication (cost of sending an SMS). In addition to that, research has proven SMS to be one of the least secure authentication factors [39]. Attacks such as SIM swapping, SS7 flaw exploitation, and GSM traffic eavesdropping and interception make it difficult to trust this method. Services managing crucial personal information, such as banking, are not recommended to use SMS as an authentication factor. Other services that do not store sensitive data can use SMS, even though other methods are still preferred. Nevertheless, if no other methods are available, SMS as a second factor is still advocated for, not SFA.

NIST also classifies SMS-based authentications as a restricted authenticator, recommending the following:

- Offer users at least one alternative unrestricted method;
- Mention the associated risks to users;
- Create a migration plan in case SMS is no longer acceptable at some point.

2.6.4 Single-Factor OTP Device

This type of authenticators, either hardware (e.g. YubiKey, RSA SecurID) or software-based (e.g. Google Authenticator, Microsoft Authenticator), are capable of generating one-time passwords (OTP) using an embedded secret (seed) shared between the OTP device and the server. This secret can either be a time-based nonce (TOTP) or from a counter present on both sides. After computing, the password is displayed on the device. Then, users must transmit it to the server, proving possession and control of the device. Therefore, it is considered a "Something you have" factor.

There are some differences between the two types of OTP devices. Time-based devices must maintain an internal clock that is synchronised with the server clock. This is particularly important for hardware-based TOTP, as most software-based TOTP (mobile authenticator applications) can synchronise clocks through the internet or mobile network. Even though hardware devices are

usually shipped with pre-synchronised clocks, they naturally drift apart. As such, servers need to consider possible clock drift when determining the validity of an OTP value. However, as time drifts average around 2 minutes per year, it is natural these devices need a replacement after some years [40].

Regarding non-time hardware-based OTP devices, typically, it generates a password based on pressing a button. As such, users may press the button several times, either by mistake or as a test. So, servers must accept any of several possible future passwords and advance their current state to the device state.

2.6.5 Multi-Factor OTP Devices

These devices are similar to Single-Factor OTP Device, with the addition of requiring activation by the input of a memorised secret or by the presentation of a biometric. It is classified as a "Something you have" factor, activated by a "Something you are" or a "Something you know" factor.

2.6.6 Single-Factor Cryptographic Software

This secret cryptographic key and associated software are stored on a software-accessible medium. It is classified as a "Something you have" factor. An example is the use of a client X.509 certificate. The certificate is accompanied by a private key held by the user. So, to associate a certificate with a user, the certificate is signed by a certificate authority the server trusts (sometimes signed by the server itself). The certificate "common name" field is typically populated with the user identity on the server.

To be classified as a Multi-Factor Cryptographic Software, activation of the authentication factor would be required, either through a memorised secret or a biometric characteristic. Each authentication operation would require the activation factor, which must be separate from unlocking the host device.

2.6.7 Single-Factor Cryptographic Device

Similar to Single-Factor Cryptographic Software, except that the private key is contained within a hardware device. It is classified as a "Something you have" factor.

Examples of these devices are Smart Cards (e.g. Portuguese citizen card), FIDO U2F keys (e.g. YubiKeys), or devices with hardware trusted platform modules (TPM) such as modern computers and smartphones [31].

Organisations should consider the methods' capabilities, as some of the mentioned devices operate in multiple modes and may have more stuff than others.

The authentication factor must be activated to be considered a Multi-Factor Cryptographic Device, as described in Single-Factor Cryptographic Device.

2.7 Summary

This chapter provides an in-depth overview of contemporary tools and protocols used in the domain of authentication and identity management, emphasising the technical architecture, standards, and security mechanisms that underpin modern digital identity systems. The chapter begins by introducing the authentication and authorisation protocols, SAML 2.0, OAuth 2.0 and OpenId Connect 1.0. SAML 2.0 is a standard for exchanging authentication and authorisation data between identity providers and service providers. On the other hand, OAuth 2.0 is highlighted as a framework that allows third-party applications to access user resources without exposing credentials, ensuring user consent and security. Its extension, OIDC, provides an additional layer of identity verification on top of OAuth, enabling decentralised authentication. Together, these protocols facilitate secure delegation and authentication for both web and mobile applications.

Further discussion covers LDAP, a protocol for accessing and maintaining distributed directory information services over a network. LDAP is extensively used for managing user identities, particularly within corporate environments, where centralised authentication is required for managing access to network resources.

In the realm of cryptographic authentication, WebAuthn and X.509 client certificates are examined. WebAuthn, part of the FIDO2 standard, offers passwordless authentication using public-key cryptography, improving security and user experience. X.509 client certificates, widely used in TLS/SSL-based authentication, provide a robust framework for mutual authentication by verifying the identities of both clients and servers using PKI.

The chapter also presents Autenticação.Gov, the national identity authentication system of Portugal, which leverages public-key cryptography and secure MFA to provide access to a wide range of government services. This system integrates with European Union identity frameworks, exemplifying cross-border interoperability.

Then, the chapter introduces two existing IdM solutions. First, Shibboleth Identity Provider, an open-source solution widely used in federated identity management systems, particularly within academic institutions. Shibboleth supports SAML, enabling secure Single Sign-On (SSO) across different systems and organisations. Then, the chapter explores Keycloak, a comprehensive identity and access management solution that supports multiple protocols such as OAuth 2.0, OIDC, and SAML 2.0.

To conclude, the chapter discusses the NIST security guidelines, particularly those concerning authentication factors and the principles of MFA. NIST's guidelines underscore the need for strong authentication mechanisms, recommending a combination of knowledge-based, possession-based, and biometric factors to mitigate the risk of credential compromise. By following these guidelines, organisations can enhance their security posture while ensuring compliance with global best practices.

In summary, this chapter consolidates key tools, protocols, and standards that shape the current landscape of identity management, offering a comprehensive guide to both foundational and cutting-edge technologies.

Chapter 3

University of Porto AAI

Similarly to other universities, U.Porto provides its students and staff with services and resources (e.g. computers, printers, websites, VPN, etc.) accessible using its authentication systems. Depending on each resource's characteristics, some disparities may be found regarding their operating system, age, location, supplier, hardware, etc.

This, in turn, provides a challenge, as U.Porto's AAI must guarantee compatibility with all possible resources. For example, students may access resources using Linux or Windows and different operating system versions. The AAI must be compatible with all these possible versions of operating systems. Furthermore, as cybercrime grows, so do the security standards applied to authentication systems.

Besides the previous points, when it comes to permissions management, universities are a special type of entity. Every academic year, U.Porto has upwards of 4,500 new students. At the same time, it loses students that have graduated. On top of that, students may enrol, drop or complete a course any day. This means adding 4,500 new users to its databases, revoking privileges to those who have completed classes, leaving the university, or giving privileges to others.

This chapter will present the state of U.Porto's authentication system. It starts by focusing on the existing user directories, followed by the IdM software used to authenticate the users. Afterwards, the focus shifts to the limitations of the AAI and the requirements imposed by federations of which U.Porto is part of the composition. To finalise, the current flows of authentication and their limitations are presented.

3.1 Directories

U.Porto stores its users' information in directory systems such as OpenLDAP or AD. These directories are the basis of the entire authentication system, either by directly authenticating a user when accessing, for example, a virtual private network (VPN) or computers or by providing authentication to higher-level services. These directories contain user data, credentials, groups, and

privileges. They can be divided into 3 categories: central directories, constitutive entity directories (faculties and autonomous services), and department directories. The focus throughout this chapter will be on the first type, as there are plans to terminate the remaining directories.

3.1.1 OpenLDAP

There are currently two central OpenLDAP directories at U.Porto. The first one, OpenLDAP U.Porto, was created 15 years ago to support federated authentication and other services such as eduroam. The service provides redundancy by replicating every change from the master to the slave. In addition, when this service was created, a decision occurred to separate the faculty's directories, starting a problem early on. For example, imagine a student enrolled in faculty A but also attending classes in faculty B. This student must have an entry in faculty A and B directories, a clear case of inefficiency.

The goal is to replace OpenLDAP U.Porto with the other central directory: OpenLDAP Campus. Contrary to its predecessor, it unifies every faculty, meaning a student only has one entry, even if he attends classes in several faculties. Also, it provides redundancy, with the existence of three master-slave pairs, one for each data centre in the U.Porto.

Another important part of the OpenLDAP used are schemas that define relevant attributes to information exchange between academic institutions. The U.Porto currently makes use of Schema for Academia [41] (SCHAC) and eduPerson [42]. Moreover, OpenLDAP is crucial for compatibility with Linux systems inside the U.Porto.

3.1.2 Active Directory

In the context of the U.Porto, Active Directory (AD) guarantees compatibility in the Microsoft ecosystem, which is a big requirement for the U.Porto, given many of its computers inside the different faculties execute Windows Operating System.

In the same way as OpenLDAP, initially, there were directories for each faculty, named AD Campus. Naturally, there are intentions to replace AD Campus with a new service, AD U.Porto, reducing the final number of directories, if possible, to only one.

3.2 Federations

The U.Porto is part of a Portuguese higher education federation, the Science, Technology and Society Network (RCTS). This federation provides its members with advanced digital services [43] to develop, integrate and access international research projects and resources. Some of these services are:

- The online knowledge library [44] (b-on), which makes available to students some of the main providers of scientific articles;
- Filesender, which is a secure way of temporarily sharing files with any person;

- Colibri, a video conference service.

To be a part of RCTS, the U.Porto must comply with at least one requirement: use SAML 2.0 to communicate with the RCTS AAI. The federation even recommends using Shibboleth or SimpleSamlPHP as IdP for its members, providing documentation to support its installation and configuration.

By being part of RCTS, the U.Porto was able to integrate the eduGain [21] federation. This federation is a worldwide federation, providing services to nearly 27 million students, researchers, and educators. Similar to RCTS, this comes with a set of requirements. Coincidentally, eduGain also requires its members to use SAML 2.0.

3.3 Identity Providers

U.Porto's AAI is comprised of two IdPs: Shibboleth and KeyCloak.

Shibboleth is mainly used for SAML2 authentications and integration with national and international federations and Keycloak is used to provide authentication services to web applications, mobile applications and API consumers that support the OIDC protocol.

3.4 Summary

This chapter illustrated some of the challenges the U.Porto faces regarding identity management. Regarding requirements, there is an emphasis on the use of SAML 2.0, and the use of the schemas for LDAP to facilitate exchanging information with other entities. At last, it is identified that the system already uses two state-of-the-art tools.

Chapter 4

Proposed Solution

This chapter presents a recap of the problem statement, followed by an overview of the U.Porto requirements. Afterwards, the prototype's architecture components are outlined, explicitly defining the frameworks, authentication factors and protocols to implement.

4.1 Problem statement

As previously outlined in the preceding chapter, the university currently employs two distinct authentication flows - Auth.Gov and username/password-based authentication. While the Auth.Gov flow is an MFA flow, this flow is never enforced. Users may access all of the university's services using the username/password flow. It is evident there exists a discrepancy between the present-day authentication security requirements and the current U.Porto's AAI.

This may result in security incidents, which can impact the reputation of U.Porto, its community and its associated organisations, such as research centres and faculties, and may lead to the theft or loss of valuable assets, such as business information or intellectual property.

Consequently, this issue may produce information security incidents which may ultimately lead to financial losses for the university. Indeed, surveys have demonstrated a correlation between the leakage theft of credentials and data breaches [45]. An incident of this nature could have a significant detriment on U.Porto's national and international reputation, potentially give rise to litigation resulting from personal data leaks and may also result in the theft of researchers' intellectual property.

In accordance with the recommendation set forth by NIST and other renowned entities, the implementation of MFA in U.Porto's AAI would serve to enhance the security of the system, thereby protecting it from potential malicious actors. Although not a perfect solution, research conducted by Microsoft indicates that MFA has the potential to reduce the risk of account compromise by 99.22%[46]. Furthermore, the same survey suggests that even in instances where credentials have been leaked, MFA can reduce the risk by 98.56%. Consequently, the solution lies in the development of a prototype AAI supporting MFA built upon the existing U.Porto's AAI.

4.2 U.Porto requirements

As Figure 4.1 demonstrates, U.Porto's AAI is composed of two IAM tools (*Shibboleth* and *Keycloak*), two directory services (*Active Directory* and *OpenLDAP*). Furthermore, U.Porto is affiliated with the RCTS and EduGain federations, therefore it is required to adhere to the requirements set forth by each federation.

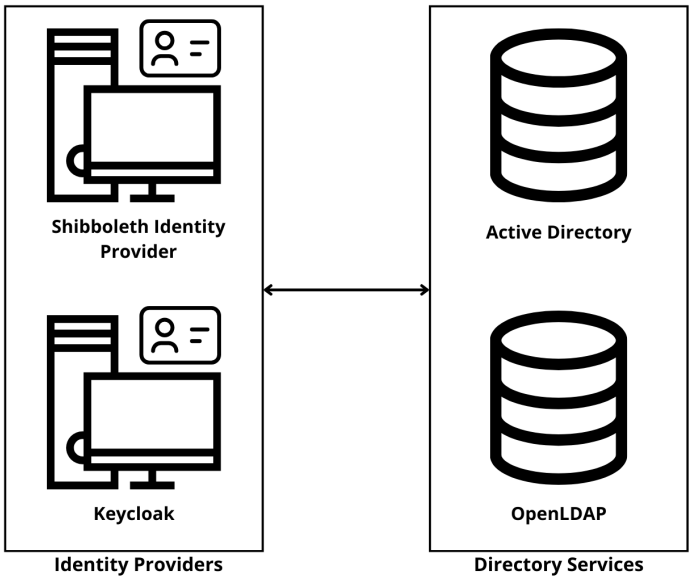

Figure 4.1: U.Porto's AAI.

Even though our project aims to improve the existing AAI, it must comply with the following obligations:

- Compatibility with the current OpenLDAP and Active Directory services;
- Abide to the EduGain and RCTS federation membership technological requirements;
- Use an SSO authentication model;
- Compatibility with existing services depending on U.Porto as an Identity Provider.

4.3 Architecture

The proposed architecture is constituted of four interconnected components, each of which is responsible for a discrete function within the solution. The relationship between the components is illustrated in Figure 4.2.

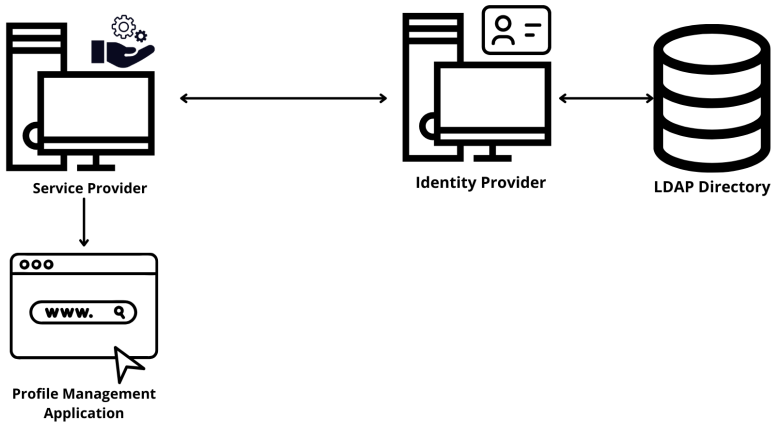

Figure 4.2: Proposed solution architecture.

The first component is the LDAP directory, which functions as the central database for the storage of user information. It is employed to facilitate authentication features and to provide users' data when requested.

The second element is the IdP, which performs authentication features and, with the assistance of the aforementioned directory service, supplies users' identities to both internal and external services.

Lastly, the third piece is a profile management application. The objective of the application is to provide users with control over their authentication, allowing them to enable, disable and configure their authentication factors.

The following subsections provide a detailed account of the preceding items and the frameworks employed. Additionally, the selected authentication protocols, authentication factors, and environment setup are also depicted.

4.3.1 Directories

As mentioned in Chapter 2, LDAP is a tool used to store and organise information about users, systems and other entities. In regards to the prototype developed, it is used to save users' data and aid in password-based authentication.

Despite the existence of numerous implementations of the protocol, only two were considered for this project: Active Directory and OpenLDAP, both of which are used by U.Porto. The former, however, is not open source software and was thus excluded from further discussion, leaving OpenLDAP as the sole remaining directory technology.

4.3.2 Identity Provider

The IdP is responsible for user authentication, enabling SSO and federation integration.

Moreover, the IdP must be capable of supporting the use of MFA through either integrated support or software customisation. Additionally, it is essential that the IdP is able to support the

integration of LDAP, as this will be the means by which it will retrieve user identity data. Finally, it is expected that the authentication flows can be customised, either through the user selecting the authentication factors to be used or through the SPs specifying the accepted authentication factors.

Regarding the tools used, the discussion was built around the two previously identified solutions: Shibboleth IdP and KeyCloak, both of which are used by U.Porto. Table 4.1 presents a comparison of the main features of both tools.

Table 4.1: Comparison of main features integrated in KeyCloak and Shibboleth IdP.

Feature	KeyCloak	Shibboleth IdP
MFA	✓	✓
TOTP Authentication	✓	✓(Partial)
FIDO Authentication	✓	X
X509 Authentication	✓	✓
IP Address Authentication	X	✓
SSO	✓	✓
SAML	✓	✓
OIDC	✓	✓
OAuth	✓	✓
LDAP	✓	✓
Customisable flows	✓	✓
Authorisation	✓	X
Ease of configuration	✓	X
Ease of deployment	✓	✓
Identity Management	✓	X

Despite the fact that the tools compared pursue comparable objectives, an analysis of the number of features reveals that KeyCloak has a slight advantage. First, the learning curve is lower due to the availability of a user administration interface for configuration purposes, whereas Shibboleth's configuration is performed by manually editing the configuration files. Additionally, KeyCloak natively supports some authentication factors that are not fully or partially supported by Shibboleth. Finally, in contrast to Shibboleth, KeyCloak is a holistic IAM solution, offering the additional capability of providing authorisation services.

Notwithstanding the aforementioned advantages of utilising KeyCloak, the decision was taken to adopt Shibboleth as the IdP software. The decision was based on the following considerations:

- Shibboleth is relatively straightforward to deploy once the configuration files have been completed;
- It is also recommended and supported by the RCTS federation, and it provides a configuration and deployment tutorial for its federation members, such as U.Porto;
- Shibboleth is a widely used tool in universities across the globe;
- In contrast to Shibboleth, which is owned by a consortium of primarily academic institutions, KeyCloak is owned by RedHat, a private company. This may result in the product

being discontinued or significantly altered, potentially rendering it unusable. Consequently, Shibboleth is regarded as a more stable option;

- It is possible to bridge between Shibboleth and Keycloak. In such a scenario, the Shibboleth IdP may provide authentication services that are consumed by the Keycloak IdP. At this point, Keycloak takes control and provides the authorisation and authentication information required by the application.

4.3.3 Profile Management Application

One of the limitations of the Shibboleth IdP is the absence of an integrated interface through which users can manage their authentication factors and their overall identity. This is due to the fact that the Shibboleth IdP software was developed with the primary objective of providing IdP functionalities, rather than IAM features. To illustrate, when a user accesses their Google account, they can manage several aspects of it, particularly the manner in which they can log in by configuring the password, recovery codes, backup emails, MFA, and so forth. U.Porto currently lacks a service through which a user may achieve these, in part due to the absence of internal authentication flows, with the exception of the password-based flow.

For the project to be fully operational and to enable users to manage their authentication factors, the prototype must include a tool that provides these functionalities. Given the scope of the project, it is imperative that the prototype allows users to configure the authentication factors associated with their university account.

This application integrates the architecture as a service, in which users are required to log in to their accounts. However, the application relinquishes the responsibility of authentication to the IdP. As the goal of the project is not to implement communication between an application, it was decided to deploy Shibboleth's Service Provider which will protect the access to the application. Given its compatibility with the IdP, comprehensible and extensive documentation, and reliable performance, the choice was seen as facilitating the work, freeing attention to the implementation of the profile management application and the configuration of the IdP.

Regarding the development of the application, the decision was taken to utilise the Flask framework, which is based on the Python programming language, and is designed for the creation of web-based software. Furthermore, Flask is also a framework frequently by UPdigital's teams. Therefore, choosing Flask as a development framework facilitates, if desired, subsequent development, issue resolution, and deployment of the application.

4.3.4 Authentication protocols

Similarly to Shibboleth's IdP, its SP is built to only use SAML2 protocol. However, in contrast with the IdP, the SP does not have any plugins which implement the use of OIDC. Therefore, the decision was to either remove the SP and use another solution that implements OIDC or use SAML2 protocol. The latter was chosen due to its convenience. Otherwise, the OIDC plugin

would need to be configured in the IdP as well as employ an OIDC solution to control access to the PMA.

As the goal of the project is to develop a prototype which, after some adjustments, could be deployed by U.Porto, a decision was taken to try to replicate, when possible, the IdP configuration used by U.Porto. Therefore, the communication from and to the IdP will be achieved through the use of the SAML2 protocol.

4.3.5 Authentication factors

Even though password based authentication does not comply with a good security standard, it was decided to implement it. First, as referenced previously, password-based authentication is currently deeply entrenched in modern-day society. As the current main method of authentication, it would not make sense to remove it without first decreasing its usage. On the other hand, Auth.Gov is an MFA authentication flow from an established third party. Given its growing adoption among the Portuguese population, the security assertion it provides, and the existing compatibility with the designed architecture, it should not be excluded from the AAI.

However, these methods do not fulfil the existing requirements. As several users may not have activated the Auth.Gov application, it would mean that for them, the only available factor would be username/password authentication, which is not enough. As such, additional factors must be implemented to improve the security of the authentication of users.

In Section 1.1.1, the types of authentication factors were identified: knowledge, possession, inherence, location, and behaviour based. Although promising, especially against bad actors and fraudulent actions, behaviour-based authentication is a new type of authentication still in development and not widely used. For these reasons, it was decided to exclude it from further consideration. Similarly, the location-based authentication factor was also discarded because, contrary to some enterprises that work solely on a specific geographical location, U.Porto users may access its services from all across the globe. An example can be researchers or professors attending events in other countries or cities, who may need to access U.Porto's internal services.

Ever since wide MFA adoption, the usage of authentication factors has severely changed, from the development of new factors and methods of authentication to the depreciation of existing ones in favour of newer ones. Surveys [38] [47] show that, after passwords, the most popular and most used authentication factors are:

1. Security questions;
2. SMS, Email or Voice OTP;
3. Software OTP;
4. Hardware OTP;
5. Push authenticator apps (e.g. Duo);
6. Biometrics;

7. Security key (e.g. Yubico);

Addressing the first factor, security questions have been widely used, most commonly used for password resets on platforms. Even though it is still commonly used as an authentication factor, NIST no longer recognises security questions as an acceptable authentication factor. Studies from Microsoft and Google have shown that easy-to-remember security questions are not secure and that secure questions are not usable [48], which defeats the point of implementing them. Therefore, this factor was no longer considered for implementation.

Regarding the second, third and fourth factors, OTP is one of the most widely used factors in 2FA due to its ease of use, which is one of the key principles for MFA adoption. Studies have shown that SMS OTP is one of the least secure forms of 2FA [39]. Additionally, there are associated costs with this kind of service, as well as voice OTP and hardware OTP, which are incompatible with the prototype development. Email OTP, on the other hand, does not have costs, but it lacks security when compared to software OTPs, which do not have major disadvantages regarding pricing, configuration or usability through free and secure mobile or desktop applications. Even though Shibboleth IdP does not natively support OTP authentication, the community and the project team have developed a plugin, currently officially supported by the development team, which adds this functionality to the IdP [49].

On top of that, several enterprises also employ authenticator apps configured to push authentication notifications requesting authentication approval to the user. Examples of this are Duo's or UserLock's push applications. Similarly to SMS or voice OTP, this type of solution typically implies costs to the project as the services would need to be contracted. The only solution would be to develop and implement a mobile app containing this feature, which would be out of the project's scope. Given these reasons, it was decided to exclude push authentication applications from the architecture.

For the last two factors, biometrics and security keys, there exists the FIDO2 standard which allows users to seamlessly authenticate on their accounts by associating their devices to their accounts. This is accomplished through public key cryptographic techniques, which classify FIDO2 as a phishing-resistant authentication factor. Additionally, it also provides MFA capabilities by itself by providing proof of ownership of the device and requesting a PIN or a biometric factor to unlock the cryptographic techniques, thus providing a true MFA. At the time of writing, Shibboleth IdP does not provide support for this technology, but there exists the development of a plugin to provide this authentication factor [50].

Similarly to WebAuthn, X.509 is also a possibility for user authentication. In fact, Stanford University developed a security initiative, Cardinal Key [51], designed to enhance the authentication process within the university's digital infrastructure by reducing reliance on traditional passwords. It employs client-side X.509 certificates, which are cryptographically tied to user identities and securely stored on users' devices. This system provides a seamless and secure authentication experience, allowing users to access Stanford's online services without repeatedly entering passwords. By integrating with existing multi-factor authentication (MFA) systems, Cardinal Key

adds an additional layer of security, ensuring that only authorised individuals can access sensitive resources.

However, while initiatives such as Cardinal Key enhance security and user convenience, they also present several disadvantages, particularly in terms of dependency on device security and potential complexities in key management. The system relies heavily on the security of the user's device, as the cryptographic keys are stored locally; if a device is compromised, lost, or stolen, the security of the Cardinal Key could be jeopardised, potentially leading to unauthorised access. For this reason, Stanford put in place minimum security requirements for all devices. Moreover, the management of cryptographic keys, including the processes for renewal, revocation, and recovery, adds complexity for both users and administrators, potentially leading to operational difficulties, especially in large-scale deployments. These factors can complicate the widespread and effective use of such tools within digital ecosystems. Therefore, it was decided to implement X.509 authentication on the project's prototype. Based on the aforementioned arguments, X.509 authentication will only be provided to a small subset of the U.Porto's users, as widespread use would require developing a higher complexity solution, which is considered impracticable for the project.

4.4 Summary

This chapter begins by reiterating the pressing problem of escalating cybercrime, emphasising the vulnerability of systems and networks to sophisticated cyberattacks. The growing incidence of phishing, credential theft, and brute-force attacks highlights the inadequacy of traditional password-based authentication systems. In response to these threats, the chapter underscores the critical role of MFA in safeguarding digital infrastructures. By requiring multiple independent credentials, MFA significantly enhances security, serving as a vital defence mechanism against unauthorised access. Within the context of the university's AAI, MFA emerges as the most effective solution for ensuring secure and reliable user authentication across institutional systems.

The chapter proceeds to propose a robust identity management solution for the university's AAI, leveraging Shibboleth's IdP, LDAP, and a custom-developed application named PMA. The IdP facilitates secure SSO and federated identity management, allowing users to authenticate once and gain access to multiple services within the university network. LDAP serves as the backbone of the university's identity repository, managing user credentials, roles, and access privileges in a centralised directory system. The PMA application acts as an interface that integrates with both Shibboleth and LDAP, streamlining user management and authentication workflows. Together, these components form a cohesive architecture designed to meet the institution's security and usability requirements.

The chapter concludes with a detailed presentation of the authentication methods selected for implementation, emphasising a multi-layered approach. WebAuthn, a FIDO2-based standard, is introduced as a cutting-edge, passwordless authentication method that utilises public-key cryptography for enhanced security and user convenience. X.509 client certificates are proposed to provide an additional layer of security through mutual authentication, ensuring that both the client

and server identities are cryptographically verified. Traditional passwords are maintained as a baseline authentication factor, supplemented by TOTP to provide an additional possession-based factor. Lastly, the chapter includes Autenticação.Gov, the Portuguese national identity solution, as a trusted external authentication provider, allowing for secure access to university services via government-issued credentials.

In conclusion, the chapter presents a comprehensive solution to address the university's AAI challenges by combining multiple modern authentication protocols and tools. By implementing MFA with diverse authentication methods—WebAuthn, X.509, passwords, TOTP, and Autenticação.Gov—the proposed architecture not only mitigates the risks of cybercrime but also ensures that the university's systems remain accessible, secure, and resilient.

Chapter 5

Implementation

This chapter outlines the process of implementing the proposed solution. It begins with an overview of the implementation environment and technologies employed throughout the project. The implementation details are then presented in two sections, each representing the IdP component and the Profile Management application, respectively. Finally, the last section describes the challenges faced and the solutions devised to address them.

The implementation of the system was conducted on two distinct virtual machines (VM).

5.1 Shibboleth Identity Provider

As previously discussed, the Shibboleth Identity Provider (IdP) was selected as the authentication provider for the prototype's architecture. This decision was informed by the robust security features, flexibility, and widespread adoption of the Shibboleth IdP in federated identity management systems. In order to deploy the IdP, the Java Servlet container software Jetty 10.0 was used. Additionally, an OpenLDAP server was installed and configured on the VM to supply directory services to the IdP.

The standard installation of Shibboleth IdP, while comprehensive, did not fully align with the unique demands of the prototype. As a result, it was necessary to undertake extensive configuration and customisation of the base software. These adjustments included the configuration of advanced authentication flows and the extension of the software to support additional authentication factors beyond the default configuration.

5.1.1 Password

Shibboleth's IdP is equipped to facilitate password-based authentication, which can be configured as an integral component of the broader SSO process. In a typical configuration, the IdP authenticates users by verifying their credentials (usually a username and password) against a backend identity store. This authentication mechanism is integrated into the Shibboleth IdP through the *authn/password-authn-config.xml* file, wherein administrators define the authentication flow and specify the credential verification strategy. The configuration enables the IdP to authenticate users

by validating their passwords against a variety of backends, including databases, LDAP directories, and Kerberos, to cite one example. The flexible architecture of Shibboleth permits the password authentication process to be tailored to align with the specific security policies and user requirements of an organisation.

In order to simulate as close of U.Porto's AAI as possible, LDAP password-based authentication was configured. The IdP utilizes the LDAP bind operation to authenticate users, where the user's credentials are verified by binding to the LDAP server using the distinguished name (DN) of the user. In order to correctly retrieve users, the attribute *idp.authn.userFilter* was configured to filter the user by the UID attribute, i.e. to match the inserted username to the user's LDAP entry UID attribute.

5.1.2 Time-Based One-Time Password

In order to enhance the security of the authentication process within the prototype system, a TOTP mechanism was integrated into the IdP. TOTP is a widely recognised method for implementing 2FA, which adds an additional layer of security by requiring users to provide a time-sensitive code generated by a trusted device, typically a smartphone, in addition to their primary credentials. Its authentication mechanism relies on a shared secret, or token seed, between the client and the authentication provider to generate time-sensitive one-time passwords.

Shibboleth's IdP provides an official extension in the form of a plugin, which facilitates the integration of TOTP in the prototype. This module is capable of generating and validating token codes in accordance with the standards set by the IETF [52]. Even though the plugin provides shell scripts to generate the seeds, it is overall unable to store or delete the token seeds. These actions are the responsibility of the Profile Management Application, which will be presented in the following section.

Shibboleth IdP allows two distinct configurations of token source seeds. While the first is a static seed, which can be configured by setting the *shibboleth.authn.TOTP.SeedSource* Spring bean to the static value. The second possibility is to configure IdP Attribute Resolver. It is of the utmost importance to ensure the security and uniqueness of each TOTP, which requires dynamic token seed across different sessions or users. The use of a static secret key would have a markedly detrimental impact on the security of the TOTP algorithm, rendering it susceptible to replay attacks and unauthorised access. Accordingly, NIST advocates the dynamic generation of secrets to guarantee that each user possesses a unique and irreproducible TOTP, aligned with best practices in cryptography security[36]. It was, therefore, decided that the authentication flow's attribute resolution step should be used to retrieve the token seeds, which by default retrieves an attribute named **tokenSeeds**. In order to facilitate the retrieval of the aforementioned attribute, it was also necessary to configure the storage of the secret key. It can thus be concluded that, given the unique nature of the secret key associated with each user, it can be considered information about the user, making it appropriate to store on the LDAP user's entry. In order to configure this attribute's resolution from LDAP, a Shibboleth attribute was defined in the Attribute Resolver to retrieve the seed from the directory entry, as presented below in Listing 5.1.

```
<AttributeDefinition id="tokenSeeds" xsi:type="Decrypted" dataSealerRef="totp.
    DataSealer">
    <InputDataConnector ref="myLDAP" attributeNames="tokenSeeds" />
    <AttributeEncoder xsi:type="SAML2String" name="tokenSeeds" friendlyName="
        tokenSeeds" />
</AttributeDefinition>
```

Listing 5.1: Attribute *tokenSeeds* definition.

Given the sensitivity of the token seeds employed in TOTP authentication, it is imperative to safeguard these elements from unauthorised access. In the IdP, this protection is achieved through the configuration of the data sealer, which is responsible for encrypting sensitive data within the system. In this case, the IdP is responsible for extracting the encrypted seed from LDAP, decrypting it and passing it to the TOTP plugin, which will use the plain-text seed to generate an OTP to validate an authentication.

Symmetric key storage and rotation is achieved through the use of a Java Cryptography Extension Key Store (JCEKS), which ensures the protection of the symmetric keys themselves and guarantees the system's resilience against potential attacks aiming to compromise the encryption process. This storage will store the keys used to encrypt and decrypt the TOTP seed. The Profile Management Application (PMA), also requires access to the symmetric keys to encrypt the TOTP seed before storing it into LDAP. Therefore, the Keystore containing the symmetric key must be shared every time a symmetric key is rotated from the IdP to the PMA.

The TOTP plugin provides a script which, by default, generates a 128 entropy bits AES (Advanced Encryption Standard) key and a JCEKS, thereby ensuring compliance with NIST recommendations [53].

5.1.3 X.509

The integration of X.509 certificate-based authentication within the IdP was initially pursued to enhance the security and user verification process. The integration of X.509 certificate-based authentication within the IdP required detailed configuration to ensure a secure and reliable authentication process.

Before any Shibboleth configurations, the Certificate Authority (CA), i.e. the central element of the X.509 authentication, was needed. A CA is an entity responsible for the issuance of digital certificates, which authenticate the identities of entities such as websites or individuals engaged in secure communications. The role of a CA is pivotal in the context of PKI by ensuring the authenticity and validity of these certificates. Typically, certificates are obtained from a trusted and reliable third-party issuer. However, the prototype was developed and implemented in an environment with access control policies configured, which severely restricted access to the environment. Consequently, there was no requirement to acquire a valid certificate from a third party. Instead, a self-signed CA was created, which will be responsible for issuing users digital certificates.

The initial configuration of X.509 authentication within the IdP was attempted using the IdP *authn/X509* login flow. In this setup, the IdP expects the user's certificate to be extracted from the browser to populate *javax.servlet.request.X509Certificate* attribute, which is essential for processing the certificate within the Java servlet environment. This, however, failed as the IdP was unable to extract a certificate and populate the required environment, thereby rendering the configuration ineffective.

In response to the challenges encountered with the previous login flow, an alternative approach was adopted using the *authn/RemoteUser* login flow. While the IdP offers several options to obtain user identity, the flow was configured to obtain a UID from the Java servlet request header, *USER_DN*. This flow leverages Apache as an intermediary to handle the certificate extraction and validation process before passing the relevant user information to the IdP.

Apache was configured with mod_ssl, a module that enables SSL/TLS support and allows for client certificate verification. While SSL is typically employed to verify server certificates, it can also be used to authenticate users. This is known as mutual SSL authentication. Additionally, the CA certificate and key were configured to ensure Apache can validate certificates. The directive ***SSLVerifyClient require*** was employed to enforce the requirement for client certificates, ensuring that the user's browser presents a valid certificate during the SSL handshake. This directive also facilitates the verification of the certificate's validity against trusted certificate authorities.

In addition to certificate verification, Apache was configured to extract the User Identifier (UID) from the certificate, which corresponds to the user's UID in the LDAP directory. This extraction was achieved by parsing the certificate's subject field and isolating the UID. Using mod_headers, Apache then sets a custom HTTP header, *USER_DN*, with the extracted UID value.

The custom header set by Apache was then captured by the Shibboleth IdP servlet during the authentication process. The IdP was configured to, through the use of the Java servlet, retrieve this header and cross-reference the UID contained within it against the UID entries in the LDAP directory. This verification step ensured that the UID extracted from the certificate matched a valid user in the system, thereby completing the authentication process. Figure 5.1 represents the flow implemented.

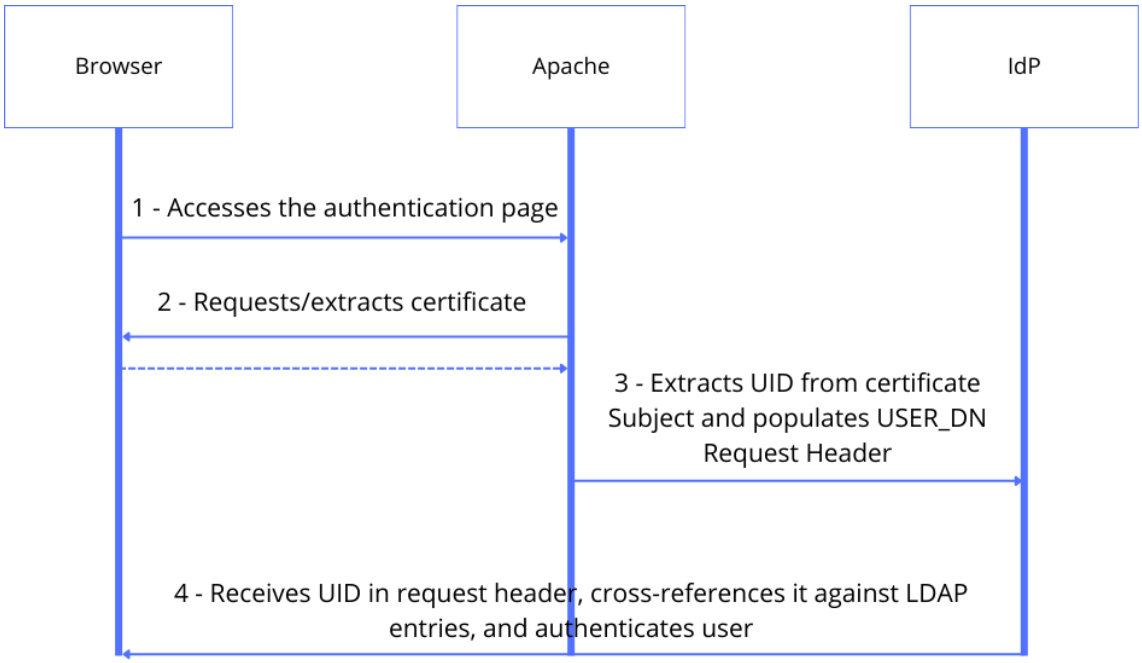

Figure 5.1: Implemented X.509 authentication flow.

To manage the certificates issued by our Certificate Authority (CA), a PostgreSQL database table was created. The database is maintained through insertions and updates to performed by the PMA, ensuring that the certificate data remains up-to-date. Figure 5.2 details a UML diagram with the structure of the table. The first attribute represents a numeric UID. The remaining attributes, except *status* and *revocation_date*, contain data about the certificate fields. The *status* attribute contains a letter representing the current status of the certificate: 'R' for revoked, 'V' for valid, and 'E' for expired. Finally, the *revocation_date* attribute contains the date of the certificate revocation, when applicable.

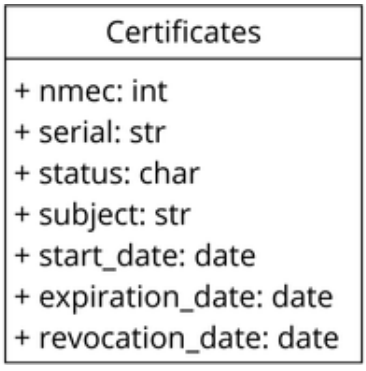

Figure 5.2: Database certificates table UML diagram.

5.1.4 Autenticação.Gov

The integration of the Autenticação.Gov (AuthGov) platform into the IdP was explored as a means to leverage the national digital authentication service for secure and standardised user authentica-

tion. To incorporate the AuthGov login flow into Shibboleth IdP, a specialised configuration was required to redirect authentication requests from Shibboleth to the AuthGov platform. This configuration involved setting up the necessary endpoints within Shibboleth to communicate with the AuthGov service and handle the authentication responses.

The login flow was designed to be seamless from the user's perspective. Upon initiating an authentication request, the user is redirected to the AuthGov portal, where they can authenticate using either their Citizen Card or Mobile Digital Key. Once authentication is successful, AuthGov returns a SAML assertion to the Shibboleth IdP, which then processes the assertion to grant or deny access based on the user's credentials.

Even though there does not exist a plugin, the source code needed for the implementation and the documented configuration steps already exist, having been developed by the University of Porto and the University of Aveiro, in conjunction with a private entity. Unfortunately, the implementation of the AuthGov login flow is contingent upon the establishment of a formal agreement with the Agency for Administrative Modernisation, I.P. (AMA)[54], the governmental body responsible for managing the AuthGov service. A protocol with AMA is required to gain authorisation to externalise authentication to the AuthGov platform.

5.1.5 MFA Flow

In the project's IdP, the configuration of MFA follows a user-driven approach, where transitions between different authentication methods are fully controlled by user actions. This approach relies on event generation, triggered by user interactions, to determine the subsequent stages of the authentication flow. The MFA configuration in Shibboleth is highly customisable, allowing organisations to define policies that control the user's transition between authentication factors based on various contextual or environmental factors such as location, device type, or user roles.

The MFA process begins when the user attempts to authenticate by submitting credentials through the Shibboleth IdP. Depending on predefined criteria or policies, the user's action triggers an event that guides the system to select the next step in the authentication flow. For instance, after providing the primary factor (e.g., username and password), the user's actions—such as confirming their identity via a second factor like an OTP or biometric verification—trigger a transition event. The system evaluates these events based on the MFA policy to determine whether additional authentication factors are required or if the authentication process is complete. Relying parties can request MFA by setting SAML 2.0 AuthnContextClassRef attribute to value *https://refeds.org/profile/mfa*. This attribute will populate the IdP AuthenticationContext, which, whenever a sub-flow, such as a Password or WebAuthn, is complete, will verify if the authentication requirements have been completed. If not, users are forced to choose a second factor and complete the authentication. Figure 5.3 illustrates the MFA flow process and the several possibilities.

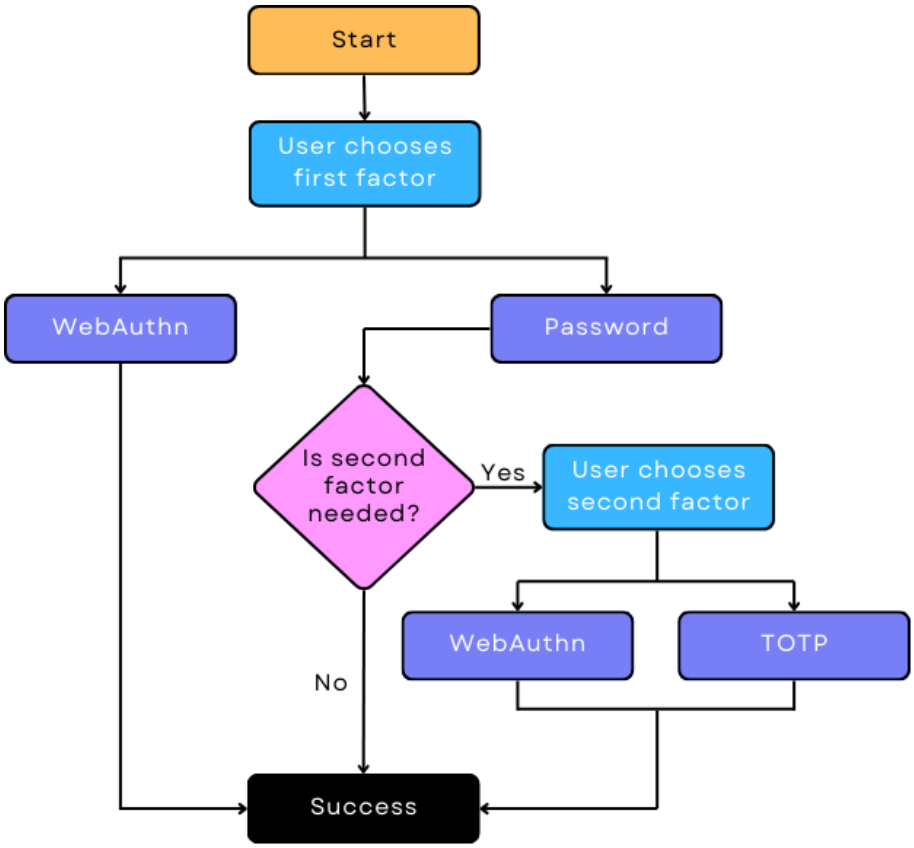

Figure 5.3: MFA Authentication Flow.

The configurations of the events used to signal the IdP are achieved in These flexible transitions are defined in the file ***conf/authn/authn-events-flow.xml***. There, we can define events, for example, *ChooseWebAuthn*. Despite this, events still need to be triggered by user actions. Listing 5.2 presents the code needed to insert a button on the web pages that trigger the aforementioned example event.

```
1 <button class="accordion-button collapsed" type="submit" name="
    _eventId_ChooseWebAuthn">
```

Listing 5.2: Example event generation button.

During the flow, we also implemented some additional validations. For example, before proceeding to handle the responsibility for the TOTP flow, verify if the user has configured its TOTP, forcing the user to re-choose the authentication factor.

5.1.6 FIDO

At the start of this project, FIDO was already being considered by Shibboleth's development team as a possible addition. However, development of this plugin has since started for the new release

of Shibboleth's IdP 5 and is currently still in active development, but its *alpha* version is available as a plugin for Shibboleth IdP v5[50].

In order to implement FIDO for Shibboleth IdP v4, which is the version used in this work, it was required to develop extensions to the base IdP software. However, in 2019, Duke University implemented WebAuthn on their AAI, which used Shibboleth's IdP[55]. While it would be possible to use this solution and implement it in this project since it has been released on GitHub [56], it has a few issues. First, it has outdated dependencies and libraries. Since its development in 2019, a new version of the WebAuthn standard has been released. Second, even though it uses a WebAuthn library from a renowned third party (Yubico), this library is not FIDO-certified or conformant. Finally, Duke's implementation does not fulfil the project's requirements. Despite all of its limitations, it was decided to use Duke's work as a basis for this thesis, mainly the way it integrates the developed code with the IdP.

As mentioned, Duke's WebAuthn implementation made use of outdated libraries for the WebAuthn standard implementation, namely Yubico's client-side and server-side implementations. For this project, however, it was decided to replace these libraries with WebAuthn4J [57] and SimpleWebAuthn/browser [58] for server-side and client-side operations, respectively. WebAuthn4J was decided due to its continuous maintenance and because it is FIDO-CONFORMANT. On the other hand, SimpleWebAuthn/browser was used because the same library is utilised on the PMA WebAuthn client-side operations. An issue arose regarding incompatibilities between WebAuthn4J 0.22.1 dependencies and between Shibboleth IdP v4.3.1 libraries, namely the Jackson Java libraries. While the IdP uses v2.14.1 of Jackson's library, WebAuthn4J utilises v2.16.1. This difficulty was quickly resolved by replacing the IdP Jackson libraries with their v2.16.1 version.

Another difference between Duke's code implementation and this project's implementation is in regard to the features. In Duke's case, their developed code can register new WebAuthn credentials from devices and allows users to authenticate using those registered credentials, which is pretty simple. On the other hand, in this thesis, WebAuthn can be used as a first (and only) or second authentication factor. In addition, it also possesses the usernameless login feature through the use of resident keys.

In order to manage the registered devices and WebAuthn credentials, a table was created on the PostgreSQL database. Deletions and insertions to the DB will be achieved by the PMA, while the IdP will mostly perform data retrieval. Figure 5.4 details a UML diagram with the structure of the table. The first attribute, *nmec*, represents the user UID number. The second to tenth attributes contain WebAuthn data. Additionally, *registration_date* and *last_authn_date* represent, respectively, the credential storage date and the last authentication where the credential was used. Finally, the last two attributes contain serialised data needed by the WebAuthn4J library.

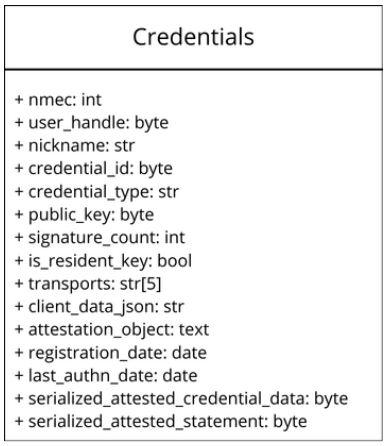

Figure 5.4: Database credentials table UML diagram.

The following list presents the Java classes developed for the project. Classes tagged with an asterisk were based on existing classes from Duke's work.

- **AttestationStatementEnvelope** - Allows the serialisation of the WebAuthn AttestationStatement;
- **AuthenticationInitiator** - Generates a PublicKeyCredentialRequestOptions when FIDO is used and second factor;
- **AuthenticatorAssertionResponseEnvelope** - Allows the serialisation of the WebAuthn AuthenticatorAssertionResponse;
- **AuthenticatorAuthenticationAssertion** - Serialises a variable, RequestId, used when FIDO is utilised as the first factor;
- **AuthenticatorCredentialAuthResponse** - Maps WebAuthn AuthenticatorAssertionResponse;
- **DatabaseConnector** - Connects and executes operations on the PostgreSQL;
- **IsSecondFactor** - Evaluates whether the authentication flow corresponds to the use of FIDO as the first or second factor in the authentication flow;
- **PopulateWebAuthnAuthenticationContext** - Creates and populates a WebAuthnAuthenticationContext class item;
- **PublicKeyCredentialRequestOptionsCreator** - Creates and configures a PublicKeyCredentialRequestOptions item;
- **RegisteredCredential*** - Maps the PostgreSQL database FIDO table to Java;
- **RegistrationServlet*** - Java servlet which handles the start of the authentication when FIDO is used as the first factor;

- **RegistrationStorage*** - Auxiliary class that serialises and deserialises items for database storage and extraction;
- **ValidateWebAuthnAuthentication*** - Java servlet which validates authenticators authentication assertions;
- **WebAuthnAuthenticationContext** - Holds information about the authentication context, such as the user's username, if the authentication is an MFA authentication, if FIDO is used as the second factor, and information of the RP.

Figure 5.5 represents the configured WebAuthn flow in the IdP, and can be described by the following:

1. Class PopulateWebAuthnAuthenticationContext is executed and creates a WebAuthnAuthenticationClass instance containing the RP ID, the RP origin and a boolean variable to know if it is MFA. Additionally, if the user previously successfully completed an authentication flow, the instance also stores the username of the user;
2. Class IsSecondFactor checks if there are any previous successful authentication flows. If so, the WebAuthn flow executing is a second-factor authentication;
3. User starts the authentication process;
4. Class AuthenticationInitiator, with the assistance of the PublicKeyCredentialRequestOptionsCreator and the WebAuthnAuthenticationContext, creates a WebAuthn PublicKeyCredentialRequestOptions (PKCRO) for a second-factor authentication. This object contains the user's stored credentials and sets the *UserVerification* to 'discouraged'. Finally, the object is sent to the user to proceed with the authentication;
5. Java Servlet instance of RegistrationServlet class receives and processes the user's authentication request and evaluates if the user filled the username input;
6. The servlet instance continues the authentication request process and, with the assistance of PublicKeyCrendentialRequestOptionsCreator, generates the PKCRO for the corresponding WebAuthnAuthenticationContext. As the username was not empty, the authentication is not a passkey authentication;
7. The servlet instance continues the authentication request process. Due to the absence of a username, the PublicKeyCrendentialRequestOptionsCreator generates a passkey PKCRO, which is then sent to the user;
8. User receives the PKCRO, and the device produces an AuthenticatorAssertionResponse, which is sent to the server;
9. ValidateWebAuthAuthentication class instance receives the device assertion and validates it. If everything matches, the authentication flow is successfully completed and an *shibboleth.AuthenticationResult* is built.

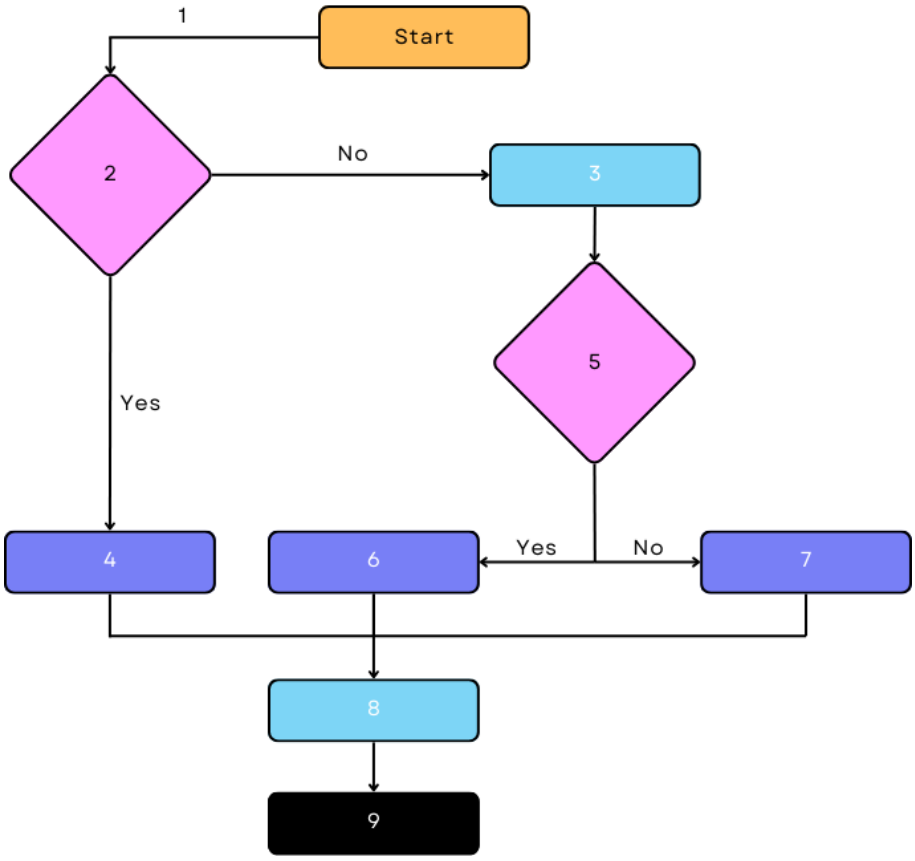

Figure 5.5: WebAuthn Authentication Flow.

5.2 Profile Management Application

The Profile Management Application (PMA) is a Flask-based web application designed to allow users to manage their authentication factors comprehensively. It is deployed through an Apache Web Server, with identity and authentication services handled by a Shibboleth SP, which integrates with the IdP described in Section 5.1. Acting as an intermediary between the PMA and the Shibboleth IdP, the SP provides single sign-on (SSO) capabilities and ensures secure authentication. Upon successful authentication via the IdP, users are redirected to the PMA, where they can manage their profile authentication.

The PMA is served via Apache Web Server 2.4.52 on Ubuntu, ensuring reliable hosting and efficient resource management. Apache was chosen due to its compatibility and integration with Shibboleth's SP. In order to achieve this assimilation, an Apache module *mod_shib_24* was used, as recommended by the SP documentation [59]. Using this module, it is possible to require the SP to provide a valid authentication session or specify the desired SAML2.0 AuthnContext class reference used during authentication, for example.

In order to centralise storage, the PMA requires access to the LDAP directory and to the PostgreSQL database, which is achieved by using, respectively, the python-ldap and psycopg Python libraries.

5.2.1 Password

The password authentication flow on the solution remains a principal factor in the authentication process. Currently, U.Porto possesses a simple platform, self-id, through which users can reset and change their passwords. Consequently, this application can integrate those features into a unified solution. Despite the simplicity of this feature, it presented an opportunity to enhance the existing one. Furthermore, the feature could be enhanced and brought up to date with the latest standards. A comparison between U.Porto's implementation and the recommendations identified in Subsection 2.6.1 revealed two potential avenues for improvement.

First, the implementation of a password strength meter when selecting a new password, thereby providing users with real-time assistance in strengthening their passwords. Despite the fact that the inclusion of special characters in passwords is often perceived to make them more complex, research suggests that the most effective method for enhancing password security is to increase their length. This is one of the reasons why the implementation of a password strength meter could prove beneficial to users. In order to implement the aforementioned feature, the TypeScript library zxcvbn-ts [60] was used. This library is based on the Dropbox library, zxcvbn, last updated in 2017 [61]. The library estimates a password's strength by means of pattern recognition and analysis of over 40,000 common passwords. Additionally, it also matches and filters out common names and popular words and recognises common patterns. Given that U.Porto is a Portuguese University, it was resolved that the Portuguese dictionary should be used in conjunction with the English dictionary, both of which are provided by the library. While this feature does not force users to choose passwords that are more resilient, it provides real-time feedback on the input of users and may assist them in making a more secure choice.

Furthermore, in accordance with the recommendations set forth by NIST, it is advisable to conduct a comparison between the proposed new password and a list of previously breached passwords. This was achieved through the utilisation of the HaveIBeenPwned Pwned Passwords feature [62]. The platform offers a variety of services, including password and e-mail lookup, designed to help individuals and organisations check whether their personal data has been compromised in data breaches. Additionally, the initiative provides an API that enables the integration of its services into the PMA. To safeguard passwords during Pwned Passwords requests, the API only requires the first five characters of the password's SHA-1 or NTLM hash, returning complete hashes whose initial five characters align. If the password is identified as having been breached, the server refuses the operation and compels the user to select a distinct password.

5.2.2 TOTP

The implementation of TOTP enables users to securely configure and revoke their TOTP devices. A single user is permitted to possess a maximum of one device. Upon commencing the configuration process, a 20 byte random secret key is generated, ensuring 160 bits of entropy in compliance with the prevailing security recommendations. Using the *cryptography* Python module[63], a

cryptography.hazmat.primitives.twofactor.totp.TOTP object containing the generated secret is created. This object is then employed to generate a provisioning Uniform Resource Identifier (URI) that adheres to Google's Key-Uri Format[64]. This process involves encoding the secret key along with supplementary metadata such as the user's account name and the organisation name. Subsequently, the URI is then transformed into a QR code through the *qrcode* Python module[65]. This facilitates the straightforward scanning of the code by users with the aid of an authentication app, such as Google Authenticator or Authy, for the purpose of configuring their device.

In instances where users require a plaintext version of the secret key, encoding is essential, as the raw bytes are unintelligible. The secret is encoded in Base32, which was chosen over Base64 due to its smaller character set, thereby reducing the probability of manual input errors by users. To finalise the configuration, users must input a 6-digit TOTP code generated by their authentication application. This step serves to confirm that the user's device is correctly synchronised with the server's TOTP setup.

In regard to the storage of secret keys, two instances of key storage may be identified. First, keys are temporarily stored in the user's application session between the generation and the confirmation of synchronisation. This is motivated by the necessity for the server to generate a TOTP token, which requires convenient access to the secret key. Once this step is complete, the secret key is cleared from the session instance. The second instance of key storage occurs immediately thereafter, wherein the key is stored in the user's LDAP entry, populating the *tokenSeeds* attribute.

It is recommended that secret key encryption be employed prior to storage, either in LDAP or in the session. In this phase, some challenges were encountered due to Shibboleth's IdP DataSealer class, which is responsible for encryption and decryption within the IdP. The class has a strictly defined encryption/decryption process [66], which includes, but is not limited to, encrypting the data using AES Galio Counter Mode (AES-GCM), with a random 16 byte nonce, as well as base64 encoding the encrypted data. Therefore, it is necessary for the PMA to perform the same encryption process as the IdP. Otherwise, the IdP will be unable to decrypt the *tokenSeeds* attribute and authenticate users with the TOTP flow. Data encryption and decryption is achieved by using the symmetric key shared with the IdP. JCEKS management is accomplished by using the *jsk* Python module.

5.2.3 X.509

The PMA provides users with the functionality to manage their X.509 certificates, including consultation, configuration, and revocation. To ensure proper security and identity verification, users are limited to one active certificate at any given time. Additionally, the generated certificates are configured to expire after 12 months. The *cryptography* Python module was utilised to perform all necessary operations on the certificates.

Users can configure an X.509 certificate by submitting a Certificate Signing Request (CSR) through the application. To maintain accuracy and consistency between the user's attributes and the generated certificate, the fields in the CSR must match the user's attributes stored in the LDAP directory. To facilitate this process, the PMA receives the user attributes, such as organisation unit,

common name, UID, and emailAddress, from the IdP in the authentication information. Furthermore, it reads the CA certificate attributes such as country, state, locality and organisation and, aggregates it with the received user data and presents it to the user within the interface. Additionally, the application provides a CSR template that users can use as a guideline for generating the request.

Once the CSR is submitted and validated, the application generates the certificate, marks it as active in the system, and stores its details in the PostgreSQL database. Simultaneously, the certificate's serial number is saved into the user's LDAP entry in the *serial* attribute, ensuring proper identification of the certificate within the user's profile. After a successful certificate generation, the application allows the user to download the certificate. Once the user downloads the file, it is removed from the server.

The PMA contains three other CSR validation steps. First, it verifies if the user already possesses a valid certificate. If so, then any request from the user is denied. Second, it verifies the CSR key type and size, denying the generation if the key is a DSA key, an RSA key with less than 2048 bits or an Elliptic-curve key with less than 224 bits. This is because NIST considers these insecure [53]. The last step validates if the serial number is a unique number in the database. Only then can a certificate be created and signed.

When a user chooses to revoke an active certificate, the PMA updates the status of the certificate in the PostgreSQL database, marking it as revoked and recording the revocation date. This ensures a clear audit trail of when the certificate was revoked. To maintain consistency, the application also removes the serial number from the *serial* attribute in the LDAP entry, signalling that the user no longer has an active certificate.

5.2.4 WebAuthn

A WebAuthn registration process follows the flow illustrated by Figure 5.6. The start initiates the process by providing to the server a nickname to the device about to be registered. Then, the server generates a PublicKeyCredentialCreationOptions (PKCCO) object, which contains a challenge to be signed by the authenticator, user information (such as a unique user byte identifier), and server metadata, and provides the PKCCO object to the browser. The browser, often referred to as the Client in WebAuthn terminology, sends the object into the authenticator, which creates a private-public key pair, signs the server challenge, stores metadata about the newly created credential on its memory, and creates an attestationObject. The server then receives an AuthenticatorAttestationResponse containing the created public key, credential metadata, and the authenticator's attestation assertion.

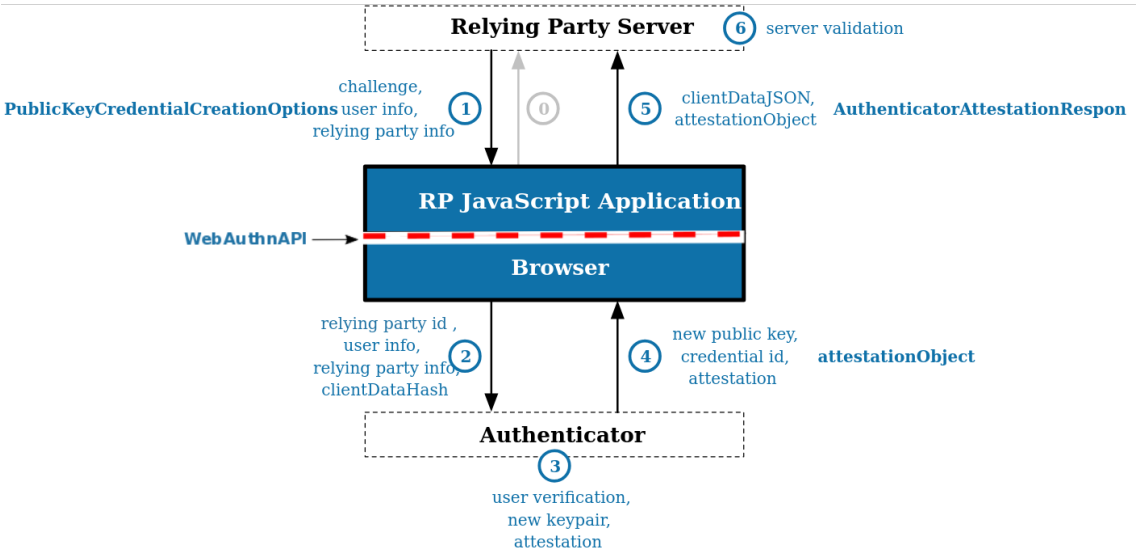

Figure 5.6: WebAuthn registration process.[1]

In order to implement such a flow, the *webauthn* Python module was used on the back end, while the *SimpleWebAuthn/browser* Javascript library was implemented on the front end of the application. Before starting the registration process, users must select a nickname to identify the device and choose whether the device can be used as a passkey or not. By default, the application disables the creation of resident keys unless explicitly enabled by the user. During registration, the server also forces user verification. Additionally, the application enforces secure cryptographic practices, ensuring that the challenges used during device registration meet the minimum requirement of 16 bytes, thus complying with WebAuthn minimum security standards. To prevent devices from having multiple registered credentials at the same time, the application sets the excludeCredentials option in the PKCCO during the registration process. This guarantees that already registered credentials cannot be reused, enhancing security by avoiding duplicate entries of authentication devices.

A further aspect of the implementation is that the PMA does not employ the attestation object to filter authenticators on the basis of their manufacturer. This implies that no particular type of device or manufacturer is singled out or excluded during the registration process. However, this can be readily implemented by incorporating a white list of approved manufacturers or a black list of rejected manufacturers.

Once the server performs the requisite validation, the pertinent information regarding the authenticator, including the public key and other relevant metadata, is stored in the PostgreSQL database. Subsequently, users are able to access the registered devices via the PMA interface, which displays relevant information such as the device name and whether it is a resident or non-resident key. Furthermore, users are afforded the option to delete devices when they are no longer

[1] W3C's figure, 2021, "Web Authentication: An API for accessing Public Key Credentials Level 2", https://www.w3.org/TR/webauthn-2/

required to do so. Upon deletion of a device, its corresponding entry is removed from the PostgreSQL database, thereby ensuring that the authentication information associated with that device is rendered invalid.

5.2.5 Recovery Codes

In the PMA, recovery codes are implemented to provide users with an alternative means of accessing their accounts if they lose access to any of the TOTP, X.509 certificates, or FIDO authentication factors. These codes are automatically generated when a user configures one of these authentication methods for the first time. To meet security standards, particularly NIST specifications, each recovery code consists of 15 random bytes, ensuring an entropy level of 120 bits. This level of randomness guarantees that the codes are both secure and resistant to brute-force attacks. By default, the PMA generates 6 recovery codes, which are presented to the user in base64-encoded format for ease of readability and use. Users are encouraged to save these codes by either copying them or downloading a file containing all six codes for future use.

The storage of recovery codes is designed with security in mind. Therefore, the codes are hashed using the Scrypt key derivation function, which includes a 16-byte random salt to ensure computational difficulty for attackers. The hashed codes and their corresponding salts are then securely stored in the PostgreSQL database, following a table schema illustrated by Figure 5.7.

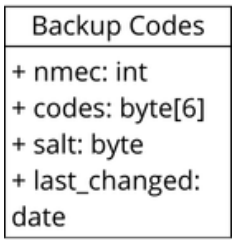

Figure 5.7: Recovery codes table UML diagram.

5.2.6 Autenticação.Gov

With regard to the implementation of Auth.Gov in the PMA, the goal was to afford users the option of either activating or deactivating the authentication factor. Accordingly, a boolean attribute was incorporated into the user's LDAP entries, with the objective of mapping the toggle of the Auth.Gov authentication flow.

5.2.7 Requesting MFA

This application handles users' security configuration and operations. It is, therefore, important to protect the application. As such, it was decided that users would be required to provide an MFA session in order to access it. Additionally, sensitive operations, such as X.509 certificate configuration, would require a re-authentication of the second authentication factor.

Subsection 5.1.5 describes the process for MFA request, in which it specifies that relying parties, such as the PMA, are required to configure the SAML 2.0 AuthnContextClassRef attribute to include *https://refeds.org/profile/mfa*. Apache, assisted by *mod_shib*, provides this functionality using the *require* directive, as exemplified by Listing 5.3.

```
1 require authnContextClassRef https://refeds.org/profile/mfa
```

Listing 5.3: Apache MFA request.

Regarding authentication elevation or re-authentication, it should be noted that Shibboleth's IdP does not possess the requisite functionality. While a solution may not be readily apparent, the IdP does, however, support the SAML 2.0 *ForceAuthn* flag within the *AuthnRequest* field. This enables the IdP to mandate a fresh authentication from the user rather than relying on the existing one. To implement this, Shibboleth's SP *SessionInitiator* was configured to initiate the authentication process. While this solution is effective, it may not be the optimal solution in terms of user experience, as users undergo a complete re-authentication using the IdP's MFA flow.

5.3 Implementation environment

The implementation of the proposed system was conducted using two virtual machines (VMs), each configured to operate on distinct versions of the Ubuntu operating system. Table 5.1 presents the characteristics of each VM. Tables 5.2 and 5.3 present the software and library dependencies of VM1 and VM2, respectively.

Table 5.1: Operating systems and computational resources of VM1 and VM2.

Metrics	VM1	VM2
OS	Ubuntu 20.04.6 TLS	Ubuntu 22.04.4 TLS
CPU	AMD EPYC 7542 32-Core Processor	AMD EPYC 7542 32-Core Processor
RAM (MB)	7962	1964
Disk (GB)	49	14

Table 5.2: Packages versions on VM1.

IdP	Apache	Slapd	JDK	Jetty	PostgreSQL	Jackson	WebAuthn4J	SimpleWebAuth
4.3.1	2.4.41	2.4.49	17.0.10	10.0.16	12.20	2.16.1	0.22.1	5.3.0

Table 5.3: Packages versions on VM2.

SP	Apache	Python	Zxcvbn	webauthn	Flask	cryptography	python-ldap	SimpleWebAuth
3.3.0	2.4.52	3.10.12	3.0.4	2.2.0	3.0.0	43.0.1	3.4.4	9.0.1

5.4 Design

Web design is a critical component of an application because it directly influences both user experience (UX) and functionality, which are key to an application's success. A well-designed interface ensures that users can easily navigate and interact with the application, improving usability and accessibility. Intuitive design reduces the cognitive load on users, making the application more enjoyable and efficient to use. Additionally, a visually appealing design can foster user trust and engagement, leading to increased satisfaction and retention.

The U.Porto design brand has been created by the UPdigital internal design team and represents a unique design concept. During the course of the project, discussions were held regarding the design. A decision was taken, and it was resolved that the thesis would be enhanced by the utilisation of a design analogous to that of U.Porto's. Consequently, the UPdigital design team was incorporated into the project with the objective of developing a design based on mock-ups for each user interface in the IdP or in the PMA. Figure 5.8 illustrates a mock-up of the FIDO devices registration pages on the PMA. This mock-up is part of a set of designs that were developed and subsequently delivered to the design team for the purpose of providing them with visual references to assist in understanding the project and its goals.

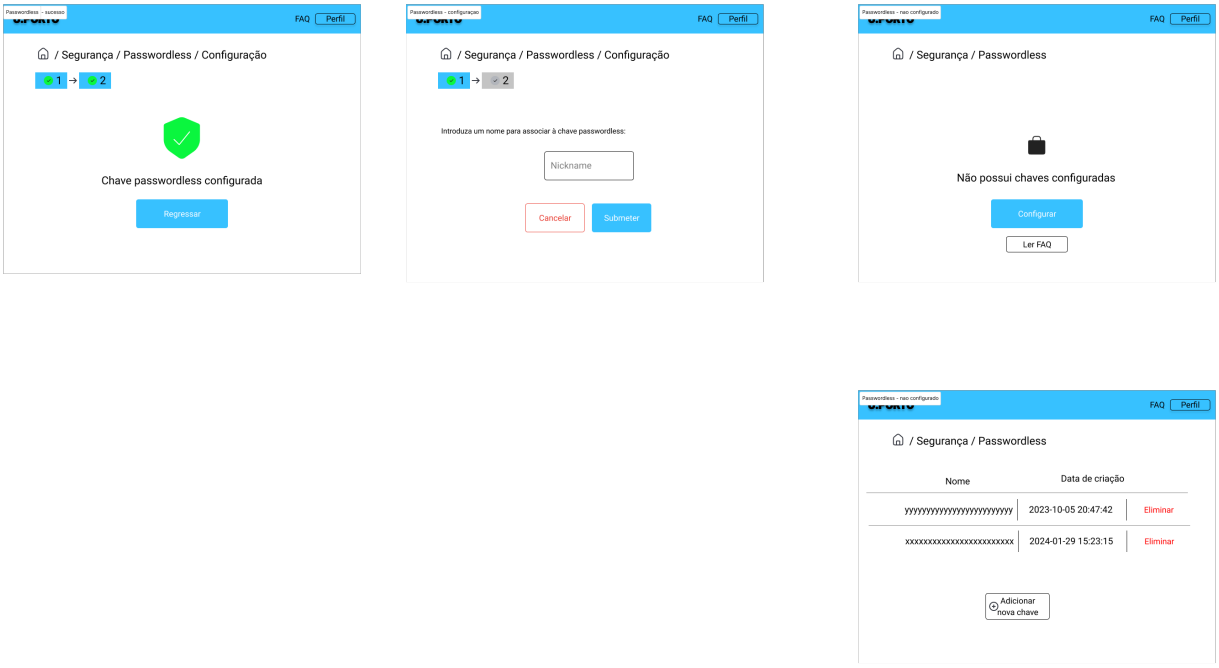

Figure 5.8: FIDO registration mockups.

The design team, comprising Carla Amaral and Vítor Carvalho, played a pivotal role in the evolution of the final design. They provided preliminary models for the PMA, as well as HTML

and CSS code for the IdP. Figures 5.9, 5.10 and 5.11 illustrate the implemented design, which is based on the design team's work and is consistent with the visual identity of U.Porto.

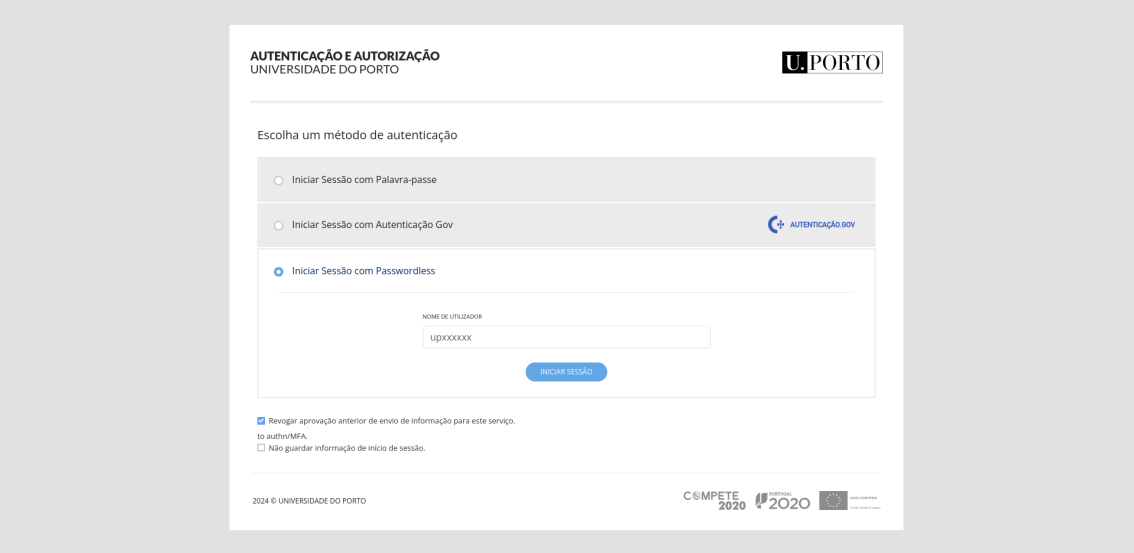

Figure 5.9: IdP first factor selection page.

Figure 5.10: IdP second factor selection page.

Figure 5.11: PMA general page.

5.5 Summary

This chapter details the implementation of the proposed authentication solution, focusing on the configuration of the IdP and the integration of multiple authentication factors. The chapter proceeds to describe the implementation of several key authentication factors. Password-based authentication is integrated as a primary mechanism, relying on the university's LDAP directory for credential verification. TOTP is configured as a second factor, providing an additional layer of security through dynamically generated passcodes. The integration of WebAuthn, a passwordless authentication method, is detailed next, with its public-key cryptographic model enhancing both security and user experience. X.509 client certificates are implemented to enable mutual authentication, particularly for use cases requiring higher levels of trust and security. Lastly, the configuration of Autenticação.Gov is described, allowing users to authenticate using the Portuguese government authentication system, providing a secure and verified external authentication source.

Following the description of these authentication factors, the chapter shifts to the development of the PMA, which is designed to facilitate user management. The application is integrated with the Shibboleth IdP and LDAP directory, enabling seamless management of user credentials and authentication workflows. Additionally, the PMA incorporates support for all the previously described authentication factors—passwords, TOTP, WebAuthn, X.509, and Autenticação.Gov—ensuring that the entire solution is cohesive and functionally interoperable.

The chapter concludes by discussing the implementation environment and the design considerations that guided the deployment. The design prioritises security, scalability, and usability, following best practices for identity management and authentication. A modular approach is adopted, allowing future integration of additional authentication factors and services as needed.

In summary, this chapter provides a comprehensive account of the technical implementation of the university's authentication solution. By detailing the configuration of the IdP, the integration of multiple authentication factors, and the development of the PMA, it offers a clear and structured guide to the deployment of a secure and scalable identity management system. The careful design

of the implementation environment ensures that the solution meets both current security needs and future demands.

Chapter 6

Conclusion

This dissertation sought to design and implement an authentication and authorisation infrastructure (AAI) that addresses the noncompliance with authentication best practices, such as multi-factor authentication (MFA) usage, at the University of Porto (U.Porto). Using Shibbleth's Identity Provider (IdP) as the core authentication service, the primary goal was to enhance authentication security by integrating several secure authentication factors in an MFA flow where users could decide the factor to use. To further support users in managing their authentication factors, a dedicated Flask web application was developed, allowing for efficient configuration and management of these security mechanisms.

The implementation successfully achieved its primary goals. The specialised MFA flow developed for the IdP ensures that users can authenticate using a combination of time-based one time passwords (TOTP), X.509 certificates, username/password combination, and fast-identity online (FIDO) devices, providing enhanced security compared to single-factor authentication. The integration of TOTP, X.509 certificates, alongside traditional password-based authentication, offers flexibility and robustness in the authentication process, catering to a wide range of security needs and preferences. Additionally, the implementation of a FIDO extension for the IdP allows users to use recent technology, which increases security and usability. The Profile Management Application (PMA) developed in conjunction with the MFA flow, enables users to easily manage their authentication factors, including password change, configuring and revoking devices and certificates, and toggling Autenticação.Gov (Auth.Gov) feature, making the solution user-friendly.

Nevertheless, several opportunities for improvement can be identified within the solution's limitations. While part of the original plan, a certificate revocation list (CRL) checking mechanism layer was not implemented. Even though the *serial* LDAP attribute is deleted once a certificate is revoked, checking its absence is not enough, because a user may have an active certificate and a previously revoked certificate. In this instance, the authentication would still succeed. As a consequence, the IdP may authenticate users providing revoked certificates, which poses a significant security risk. A potential solution could be to develop an extension which would only receive the certificate's serial and unique identifier (UID) attributes and match them with the user's Lightweight Directory Access Protocol (LDAP) entry.

Additionally, a comprehensive account recovery flow was not incorporated into the project. Consequently, users who lose access to all of their configured authentication factors may face challenges accessing their accounts. Although it may appear logical, this flow must not be implemented in the IdP. Otherwise, users could gain active authentication sessions, which would contradict the purpose of account recovery. A more secure and straightforward approach would be to develop a flow within the PMA, enabling users to re-configure their authentication factors in a limited session. It is also relevant to note that if users employ a recovery code, that code must be removed from the database.

Ultimately, the Auth.Gov login flow was not implemented. This was due to the necessity of establishing a formal agreement with the Agency for Administrative Modernization (AMA) government agency, which would entail externalising authentication requests to Auth.Gov. Consequently, it would be advantageous to pursue this agreement and integrate the existing plugin from U.Porto and the University of Aveiro on the IdP.

Bibliography

[1] Dick Hardt. The OAuth 2.0 Authorization Framework. Request for Comments RFC 6749, Internet Engineering Task Force, October 2012. URL: https://datatracker.ietf.org/doc/rfc6749.

[2] OpenID Foundation. OpenID Connect | OpenID, August 2011. URL: https://openid.net/connect/.

[3] International Telecommunication Union (ITU). Facts and Figures 2022 - Internet use. URL: https://www.itu.int/itu-d/reports/statistics/2022/11/24/ff22-internet-use.

[4] Mobile and tablet internet usage exceeds desktop for first time worldwide. URL: https://gs.statcounter.com/press/mobile-and-tablet-internet-usage-exceeds-desktop-for-first-time-worldwide.

[5] Derrick Rountree. *Federated Identity Primer*. Syngress, Boston, January 2013. URL: https://www.sciencedirect.com/science/article/pii/B978012407189600003 0.

[6] W E Burr, D F Dodson, and W T Polk. Electronic authentication guideline. Technical Report NIST SP 800-63v1.0.1, National Institute of Standards and Technology, Gaithersburg, MD, 2004. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-63ver1.0.1.pdf.

[7] Yinqian Zhang, Fabian Monrose, and Michael K Reiter. The security of modern password expiration: an algorithmic framework and empirical analysis. URL: https://www.cs.umd.edu/~jkatz/security/downloads/passwords_revealed-weir.pdf.

[8] Matt Weir, Sudhir Aggarwal, Michael Collins, and Henry Stern. Testing metrics for password creation policies by attacking large sets of revealed passwords. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 162–175, Chicago Illinois USA, October 2010. ACM. URL: https://dl.acm.org/doi/10.1145/1866307.1866327.

[9] A more secure cloud for millions of Google Apps users. URL: https://cloud.googleblog.com/2010/09/a-more-secure-cloud-for-millions-of.html.

[10] How Compromised Passwords Lead to Data Breaches & How to Prevent Them. URL: https://www.beyondtrust.com/blog/entry/how-compromised-passwords-lead-to-data-breaches.

[11] Sign in with Google - Google Account Help. URL: https://support.google.com/accounts/answer/12849458?hl=en&ref_topic=12843167&sjid=12645811706169561146-EU.

[12] Facebook Login - Documentation. URL: https://developers.facebook.com/docs/facebook-login/.

[13] What is Sign in with Apple? URL: https://support.apple.com/en-us/102609.

[14] Autenticação Gov. URL: https://www.autenticacao.gov.pt/.

[15] Cost of a data breach 2023 | IBM. URL: https://www.ibm.com/reports/data-breach.

[16] Davey Winder. Hacker Uploads 10 Billion Passwords To Crime Forum—Report. URL: https://www.forbes.com/sites/daveywinder/2024/07/05/new-security-alert-hacker-uploads-10-billion-stolen-passwords-to-crime-forum/.

[17] What is a Data Breach and How to Prevent It? URL: https://www.fortinet.com/resources/cyberglossary/data-breach.

[18] Cyber security breaches survey 2023: education institutions annex. URL: https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2023/cyber-security-breaches-survey-2023-education-institutions-annex.

[19] Estatísticas de autenticação. URL: https://www.autenticacao.gov.pt/estatisticas-de-autenticacao.

[20] OASIS. SAML 2.0 Overview. URL: https://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html.

[21] eduGAIN – enabling worldwide access. URL: https://edugain.org/.

[22] Dick Hardt, Aaron Parecki, and Torsten Lodderstedt. The OAuth 2.1 Authorization Framework. URL: https://datatracker.ietf.org/doc/draft-ietf-oauth-v2-1-08, March 2023.

[23] Final: OpenID Connect Core 1.0 incorporating errata set 1. URL: https://openid.net/specs/openid-connect-core-1_0.html.

[24] LDAP.com. URL: https://ldap.com/.

[25] FIDO Alliance - Open Authentication Standards More Secure than Passwords. URL: https://fidoalliance.org/.

[26] Understanding FIDO Standards: Your Go-To Guide. URL: https://www.okta.com/blog/2019/01/understanding-fido-standards-your-go-to-guide/.

[27] Sampath Srinivas, Drik Balfanz, Eric Tiffany, and Alexei Czeskis. Universal 2nd Factor (U2F) Overview, April 2017. URL: https://web.archive.org/web/20220817083428/https://fidoalliance.org/specs/fido-u2f-v1.2-ps-20170411/fido-u2f-overview-v1.2-ps-20170411.html.

[28] Salah Machani, Rob Philpott, Sampath Srinivas, John Kemp, and Jeff Hodges. FIDO UAF Architectural Overview, October 2020. URL: https://fidoalliance.org/specs/fido-uaf-v1.2-ps-20201020/fido-uaf-overview-v1.2-ps-20201020.html.

[29] Client to Authenticator Protocol 2.1(CTAP 2.1). Technical report, FIDO Alliance, June 2022. URL: https://fidoalliance.org/specs/fido-v2.1-ps-20210615/fido-client-to-authenticator-protocol-v2.1-ps-errata-20220621.html.

[30] Jeff Hodges, J.C. Jones, Michael B. Jones, Akshay Kumar, and Emil Lundberg. Web Authentication: An API for accessing Public Key Credentials - Level 2, April 2021. URL: https://www.w3.org/TR/webauthn-2/.

[31] Monty Wiseman. What Is The TPM (Trusted Platform Module) And Why It's Important, July 2021. URL: https://www.beyondidentity.com/blog/what-tpm-trusted-platform-module.

[32] Shibboleth. The Shibboleth Project. URL: https://shibboleth.atlassian.net/wiki/spaces/IDP4/overview.

[33] Internet2. URL: https://internet2.edu/.

[34] Project Roadmap - Development Center - Confluence. URL: https://shibboleth.atlassian.net/wiki/spaces/DEV/pages/1120895029/Project+Roadmap.

[35] RedHat. Keycloak. URL: https://www.keycloak.org/.

[36] David Temoshok. Digital Identity Guidelines. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b-4.ipd.pdf, 2022.

[37] David Temoshok. Digital Identity Guidelines. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63C-4.ipd.pdf, 2022.

[38] Okta . The State of Zero Trust Security 2022 Assessing identity and access management maturity in global organizations. Technical Report 1-888-722-7871, Okta, September 2022.

[39] Roger Piqueras Jover. Security analysis of SMS as a second factor of authentication. *Communications of the ACM*, 63(12):46–52, November 2020. URL: https://dl.acm.org/doi/10.1145/3424260.

[40] Token2 RD. Time drift: a major downside of TOTP hardware tokens, January 2019. URL: https://token2.medium.com/time-drift-a-major-downside-of-totp-hardware-tokens-c164c2ec9252.

[41] SCHAC - Standards-and-Specs - REFEDS wiki. URL: https://wiki.refeds.org/display/STAN/SCHAC.

[42] eduPerson - Standards-and-Specs - REFEDS wiki. URL: https://wiki.refeds.org/display/STAN/eduPerson.

[43] Serviços | RCTSaai. URL: https://share.fccn.pt/sites/rctsaai/4_servicos/.

[44] b-on. URL: https://www.b-on.pt/.

[45] 2024 Data Breach Investigations Report. URL: https://www.verizon.com/business/resources/reports/2024-dbir-data-breach-investigations-report.pdf.

[46] Lucas Augusto Meyer, Tom Burt, Sergio Romero, Alex Weinert, Gabriele Bertoli, and Juan Lavista Ferres. How effective is multifactor authentication at deterring cyberattacks? URL: https://doi.org/10.48550/arXiv.2305.00945.

[47] The State of Zero Trust Security 2023 | Okta. URL: https://www.okta.com/resources/whitepaper-the-state-of-zero-trust-security-2023/.

[48] New Research: Some Tough Questions for 'Security Questions'. URL: https://security.googleblog.com/2015/05/new-research-some-tough-questions-for.html.

[49] TOTP - Identity Provider Plugins - Confluence. URL: https://shibboleth.atlassian.net/wiki/spaces/IDPPLUGINS/pages/1376878877/TOTP.

[50] WebAuthnAuthnConfiguration - Identity Provider Plugins - Confluence. URL: https://shibboleth.atlassian.net/wiki/spaces/IDPPLUGINS/pages/3395321933/WebAuthnAuthnConfiguration.

[51] Cardinal Key | University IT. URL: https://uit.stanford.edu/service/cardinalkey.

[52] David M'Raihi, Johan Rydell, Mingliang Pei, and Salah Machani. TOTP: Time-Based One-Time Password Algorithm. RFC 6238, May 2011. URL: https://www.rfc-editor.org/info/rfc6238.

[53] Elaine Barker and Allen Roginsky. Transitioning the Use of Cryptographic Algorithms and Key Lengths. Technical Report NIST Special Publication (SP) 800-131A Rev. 2, National Institute of Standards and Technology, March 2019. URL: https://csrc.nist.gov/pubs/sp/800/131/a/r2/final.

[54] Integrar com o Autenticação.gov. URL: https://www.autenticacao.gov.pt/web/guest/integrar-com-o-autenticacao-gov.

[55] Shilen Patel and Mary McKee. Duke Unlock: Passwordless Authentication with Shibboleth and WebAuthn, October 2019. URL: https://incommon.org/wp-content/uploads/2020/01/2020-iam-online-webauthn.pdf.

[56] sipatel2. sipatel2/shibboleth-webauthn, April 2024. URL: https://github.com/sipatel2/shibboleth-webauthn.

[57] WebAuthn4J Reference. URL: https://webauthn4j.github.io/webauthn4j/en/.

[58] @simplewebauthn/browser | SimpleWebAuthn. URL: https://simplewebauthn.dev/docs/packages/browser.

[59] Apache - Service Provider 3 - Confluence. URL: https://shibboleth.atlassian.net/wiki/spaces/SP3/pages/2065335062/Apache.

[60] zxcvbn-ts. URL: https://zxcvbn-ts.github.io/zxcvbn/.

[61] Daniel Lowe Wheeler. zxcvbn: {Low-Budget} Password Strength Estimation. pages 157–173, 2016. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/wheeler.

[62] Have I Been Pwned: Pwned Passwords. URL: https://haveibeenpwned.com/Passwords.

[63] Welcome to pyca/cryptography — Cryptography 43.0.1 documentation. URL: https://cryptography.io/en/stable/.

[64] Key Uri Format. URL: https://github.com/google/google-authenticator/wiki/Key-Uri-Format.

[65] qrcode: QR Code image generator. URL: https://github.com/lincolnloop/python-qrcode.

[66] git.shibboleth.net Git - java-support.git/blob - src/main/java/net/shibboleth/utilities/java/support/security/DataSealer.java. URL: https://git.shibboleth.net/view/?p=java-support.git;a=blob;f=src/main/java/net/shibboleth/utilities/java/support/security/DataSealer.java;h=3e0e39fa324691321fe6a0df6925f0ab6a34a43c;hb=26261398dfaae147952f82e28892bad4aac79f39.

[67] Ken Erikson. Frameworks For Centralized Authentication And Authorization. Master's thesis, Faculty of Science and Engineering, Åbo Akademi University, 2020. URL: https://www.doria.fi/handle/10024/177327.

[68] J. Jensen and M.G. Jaatun. Federated Identity ManagementWe Built It; Why Won't They Come? *IEEE Security & Privacy*, 11(2):34–41, March 2013. URL:https://ieeexplore.ieee.org/document/6336740.

[69] Nuno Mateus-Coelho, Manuela Cruz-Cunha, and Luis Gonzaga Ferreira. Security in Microservices Architectures. *Procedia Computer Science*, 181:1225–1236, January 2021. URL: https://www.sciencedirect.com/science/article/pii/S1877050921003719.

[70] Patrick Lunney. Federation Simplified (v2). *IDPro Body of Knowledge*, 1(8), April 2021. URL: https://bok.idpro.org/article/id/62/.

[71] Rúben José da Silva Torres. Identity management: analysis of secure authentication propositions. Master's thesis, University of Porto, October 2020. URL: https://repositorio-aberto.up.pt/handle/10216/129772.

[72] OpenLDAP Project. OpenLDAP. URL: https://www.openldap.org/.

[73] Wikipedia. Active Directory, June 2023. URL: https://en.wikipedia.org/w/index.php?title=Active_Directory&oldid=1158143771.

[74] Emerson Ribeiro de Mello, Michelle Silva Wangham, Samuel Bristot Loli, Carlos Eduardo da Silva, Gabriela Cavalcanti da Silva, Shirlei Aparecida de Chaves, and Bruno Bristot Loli. Multi-factor authentication for shibboleth identity providers. *Journal of Internet Services and Applications*, 11(1):1–21, December 2020. URL: https://jisajournal.springeropen.com/articles/10.1186/s13174-020-00128-1.

[75] William Fisher. (Draft) Background on Identity Federation Technologies for the Public Safety Community. URL: https://nvlpubs.nist.gov/nistpubs/ir/2021/NIST.IR.8336-draft.pdf, June 2021.

[76] Nitin Naik and Paul Jenkins. Securing digital identities in the cloud by selecting an apposite Federated Identity Management from SAML, OAuth and OpenID Connect. In *2017 11th International Conference on Research Challenges in Information Science (RCIS)*, pages 163–174, May 2017. ISSN: 2151-1357.

[77] J. Carretero, G. Izquierdo-Moreno, M. Vasile-Cabezas, and J. Garcia-Blas. Federated identity architecture of the European eID System. *IEEE Access*, 6:75302–75326, 2018.

[78] Hakan Yildiz, Christopher Ritter, Lan Thao Nguyen, Berit Frech, Maria Mora Martinez, and Axel Küpper. Connecting Self-Sovereign Identity with Federated and User-centric Identities via SAML Integration. In *2021 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7, September 2021. ISSN: 2642-7389.

[79] Parves Kamal, Saad Mustafiz, Faisal Md Abdur Rahman, and Raihan Taher. Evaluating the Efficiency and Effectiveness of a Federated SSO Environment Using Shibboleth. *Journal of Information Security*, 06(03):166, May 2015. URL: http://www.scirp.org/journal/PaperInformation.aspx?PaperID=56785&#abstract.

[80] Asier Aguado Corman, Daniel Fernández Rodríguez, Maria V. Georgiou, Julien Rische, Ioan Cristian Schuszter, Hannah Short, and Paolo Tedesco. CERN's Identity and Access Management: A journey to Open Source. *EPJ Web of Conferences*, 245:03012, 2020. URL: https://www.epj-conferences.org/10.1051/epjconf/202024503012.

[81] Justin Richer. End User Authentication with OAuth 2.0 — OAuth. URL: https://oauth.net/articles/authentication/.

[82] auth0.com. JWT.IO - JSON Web Tokens Introduction. URL: http://jwt.io/.

[83] Michael B. Jones, John Bradley, and Nat Sakimura. JSON Web Token (JWT). Request for Comments RFC 7519, Internet Engineering Task Force, May 2015. URL: https://datatracker.ietf.org/doc/rfc7519.

[84] InCommon Federation Software Guidelines - InCommon Federation - Internet2 Wiki. URL: https://spaces.at.internet2.edu/display/federation/InCommon+Federation+Software+Guidelines.

[85] Vincent . WebAuthn Resident Key: Discoverable Credentials as Passkeys, September 2023. URL: https://www.corbado.com/blog/webauthn-resident-key-discoverable-credentials-passkeys.

[86] Universidade do Porto. Facts and Figures. URL: https://www.up.pt/portal/en/explore/about-uporto/facts-and-figures/.