Graduation Project

FIDO2 WEB PASSWORDLESS AUTHENTICATION FOR SSO SYSTEMS

Andrés Felipe Hernández León

Los Andes University

School of Engineering

Systems and Computing Engineering Department

Bogotá, 2020

Graduation Project

FIDO2 WEB PASSWORDLESS AUTHENTICATION FOR SSO SYSTEMS

Andrés Felipe Hernández León

Mentor

Sandra Julieta Rueda Rodríguez, PhD.

Associated Professor

Los Andes University

School of Engineering

Systems and Computing Engineering Department

Bogotá, 2020

TABLE OF CONTENT

FIGURES INDEX

ABSTRACT

The passwords are not completely secure anymore as the number of accounts and password-related attacks is increasing. This means that the personal or sensitive information of a user could be leaked, or it could even affect its finances as passwords are the most used authentication method. To solve this a new environment called passwordless has emerge, promising to take care of all the password related attacks as it eliminates them of the authentication process or uses multiple authentication methods in order to evade trusting fully on passwords.

Therefore, this investigation project, researched on passwordless environments and designed and implemented a passwordless infrastructure based on an educational system. Based on this design, a prototype was designed and implemented with the objective of understanding deeply this kind of environment, the reach and functionality that it may have on a real-world environment. Later, the prototype was tested and compared with the classic password-based environment to confirm its strengths and weaknesses and how the change of paradigm could mean the solution of a password-based environment.

1 INTRODUCTION

Currently, the technology has reached a point where it is needed for almost everything in our daily lives. Consequently, activities like messaging, financial transactions, investigations or leisure have migrated to an *online environment.* Moreover, most of these activities make use of personal information and other kind of sensitive data, which must be preserved with integrity and confidentiality. And historically, the first security measure used to achieve this goal is password-based protection.

This kind of protection has problems like weak passwords, the reuse of them across different accounts and data breaches that expose user's emails with its passwords. For instance, the 2020 report of Verizon points out that the credential theft and social attacks represent most of the breaches with 67% or more [1]. Moreover, the leak of a credential could cause the exposure of sensitive data of an organization, government or a person. That was the case of the Brazilian government, which in November of 2020 exposed the data of sixteen million of Brazilian COVID-19 related patients after a hospital employee published on GitHub a spreadsheet with credentials of government sensitive systems. The data contained information about the patient's name, addresses, telephone numbers and clinical conditions that include their medical history and medication regimes [2]. This incident reflects the great impact of one password leak, but ordinary people could be affected too, as a leak in their passwords could lead to identity theft, robbing bank accounts, etc.

As a result of this, a new trend arose in the market where different companies started to invest on a new environment that uses *passwordless solutions*. There, the authentication process was redesigned to accept new options; biometrics authentication and multi factor authentication became available to the users. These kinds of solutions make the interaction of the user more intuitive and easier as they do not have to remember long and strong passwords for all their accounts, neither use a password manager nor write them down somewhere (like a spreadsheet). Moreover, it is more secure as it avoids identity fraud and decreases some password related expenses like the recovery process.

However not all the companies have migrated or integrated these solutions as there is still a lot of research to do and standards to be defined. With that, the FIDO alliance was created in order to reduce the world's reliance on passwords and "*promote the development of, use of, and compliance with standards for authentication and device attestation*" [3]. This alliance has worked on distinct standards that are based on public-key cryptography and the main goal to remain simple for the final user. Two of their standards (*FIDO U2F – Universal Second Factor* and *FIDO UAF – Universal Authentication Framework)* have been used widely on the industry as they were promoted by big industries like Apple, Amazon, Google, Facebook, among others. Finally, their last standard *CTAP – Client to Authenticator Protocol* is a complement to the W3C's *WebAuthn* specification, which together are known as *FIDO2.*

Therefore, as the password-based protection is a current problem that affects not only the user's privacy, but also their money, this project intends to understand the passwordless environments. This includes its principal components, the way they work and the advantages and disadvantages that it implies. Moreover, the investigation wants to understand how the migration process from a password-based environment would be for a federated organization with multiple services.

For this, this investigation project will provide a detailed description of the passwords' problems, the previously proposed solutions and a possible solution to these problems using a passwordless environment. Furthermore, the solution contains an explanation of the passwordless environment, an overview of the protocols that involve this standard and a prototype of the solution in an educational environment. The prototype includes installation guides and an evaluation of the obtained results to confirm that the solution fulfills its objectives. Finally, some related work is presented to understand deeply the passwordless solutions and possible implementations.

2 GENERAL DESCRIPTION

Along this document several technical terms will be used. The next list provides definitions that will help the understanding of the document:

- SSO – Single sign-on: An authentication process that allows the user to authenticate itself one time for different services.
- PKI – public key infrastructure: A cryptographic infrastructure based on asymmetric cryptography, which makes use of a public-private key pair to make end to end ciphering.
- RP – Relying party: It is a server that allows access to its content or service once the user has authenticated and has a valid token given by the authentication server. In the case the user has not authenticated it redirects to the authentication server or is an intermediary in the process.
- SP – Service provider: Refers to a server that provides a service for a user, usually a RP.
- SAML – Security assertion markup language: A protocol standard between the SP and the IdP used for data exchange such as authentication.
- U2F – FIDO Universal second factor: Refers to the FIDO protocol for second factor experience.
- UAF – FIDO Universal Authentication factor: Refers to the FIDO protocol for passwordless authentication experience.
- CTAP – Client-to-Authenticator Protocol: The protocol used by FIDO 2 specifications.
- WebAuthn – Web Authentication specification: Is the W3C specification for public key credentials used in web applications.
- PRNG – Pseudorandom Number Generator: Is an algorithm that generates a sequence of “random” numbers, often used as a parameter of cryptography algorithms.

2.1 Problem description

Password protection is used as an authentication method in most of the digital sector. However, as the digital world has grown, some problems with the passwords have emerged. For instance, the use of weak passwords (short, poor combinations, common words, etc.) that can be broken with dictionary attacks, the repetitive use of these in different accounts or using similar passwords, finally, bad practices such as writing them in notes with no protection and easy access in order to remember them.

Even though, this has been a problem for years, just the last year a study conducted by google in the USA, found out that 66% of the people reuse their passwords in various online accounts, 24% use common passwords (or a variation) and 59% incorporate personal information into their passwords [4]. Also, big data breaches that expose usernames or emails with its passwords, make vulnerable all the people that reuse their passwords in multiple accounts. Each data breach exposes a lot of people as the number of records can make it to millions, as the “Collection #I” data breach that exposed nearly 773 million unique email addresses alongside passwords using various recent data breaches [5].

All these statistics create a high level of awareness as nearly half of the people surveyed by google has lost money due to data breaches, with 12% of them losing more than $500 [4]. Moreover, the data breaches include organizations accounts, that could lead to a bigger attack when the right user is targeted.

Furthermore, the increase of young users has become an unexpected problem as the kids or teenagers are not aware of the risks they take when they start an active use of the internet. As a result, the kids that count with their parent’s authorization, create accounts for games or applications they start to use, and usually the passwords are simple as they need to remember them. This kind of passwords include the family member names, important dates (i.e., birthdays) or a weak password (i.e., password, 11111111, 123456789, etc.).

Multiple research projects have proposed different methods to solve password management problems, however, these methods have been tested and failed in most of the cases. Some of these solutions were:

- The usage of aleatory generated passwords for each account that wanted to solve the password reuse problem. It failed due to the great number of different accounts that each user needs and that most of the time they forgot them and started using old passwords again.
- Password managers that take care of the passwords. This method worked for a while, but with the increase of devices and multiplatform applications, also known as the internet of things, it has become a problem for the user who needs the passwords in all the devices. Also, these managers relapse in the master password that in case of being leaked would expose all the user's personal accounts.
- Second factor authentication. Although this method improved the security of the accounts as it completely blocks unwanted access, most of the users find it troublesome when they have strict security policies that require constant authentication, thus, they rely only on passwords or do not migrate to 2FA.
- Demand the use of strong passwords and decline passwords that belong to a known data leak. This solution was used by some enterprises, but it turned out in a lot of people resetting their passwords as they forgot their secure password.

Apart from the problems previously mentioned, there are other risks related to attacks from crackers. The attacks include dictionary attacks, brute force, keyloggers, man in the middle, traffic interception and social engineering, that use different strategies that follow certain patterns. For example, the first two are based on multiple intents to guess the password and sometimes even the usernames (with knowledge of the enterprises policies to assign them). On the other hand, the other attacks try to get the user's password intercepting the data on its way to the server (Man in the middle – MIM and traffic interception) or installing malware that saves all the inputs of the keyboard (keyloggers). Finally, the social engineering attacks have different methods. First, *phishing* tries to fool users to give their credentials sending emails with links to fake websites or malicious software, and the *spear phishing,* which is an improve to the phishing as the attackers use information of the victim to make it more believable. Next, *baiting* consists in leaving infected USBs or similar storage devices near the victims hoping that they use them on their computers. At last, *quid quo pro* is an attack where the criminal pretends to be an employee in order to obtain information of the user [6].

The previous attacks can be avoided with security policies, antivirus for known malware and educational programs (in the case of social engineering attacks). Nevertheless, this does not imply full protection against these attacks as the people tend to forget the recommendations due to the increasing credibility of *phishing* mails that get better every day. Also, the antivirus cannot protect the enterprise from unknown new malware that is being developed constantly.

To sum up, passwords have proven to be insecure as there are a lot of attacks that look up to get the information and they represent a problem for the user as stronger passwords are complex and difficult to remember. Moreover, a password leak could represent great a great problem for an institution as they guard important and private information from unauthorized access. Finally, some solutions have been proposed over the years, but most of them have proven to be insufficient or incomplete for the users.

2.2 Proposal

A new solution called passwordless authentication has been gaining force on the market as this paradigm shift presents a solution to most of the problems caused by passwords. This kind of environment is based on the public-key cryptography system where the user possesses something unique for him/her that can authenticate them to anybody, in this case an identity provider. Thus, the advantage of this environment is that more than keeping something secret (the password) the user must preserve a physical object that serves as an ID on the internet, just this fact deletes all the problems presented by having a password, as for instance, a user cannot let someone else their biometrics, or would not reveal information as it is not something they know, preventing some of the social engineering attacks.

In order to understand the advantages and scope of passwordless solutions, this work has the following goals:

General objective: To design, implement and deploy a passwordless system in an educational environment. With the following objectives:

Specific Objectives:

- Understand passwordless environments, advantages and challenges.
- Design, implement and deploy a passwordless system in an educational environment.
- Analyze the decrease in the risk of identity fraud, using password related attacks, into an organization.

3 SOLUTION PROPOSAL

As the problem of passwords cannot be solved completely, the solution is a passwordless environment. Removing the passwords implies losing all the problems that they bring, without them, keyloggers, rainbow tables, brute force or social engineering attacks become pointless because of the new authentication processes. There are a lot of proposals regarding this, and they are getting stronger as the passwords become more and more insecure and expensive. Some of these solutions are unique identifiers and physical authentication mechanisms or authentication based on the user's information that do not imply memorizing something.

The unique identifiers are biometric signatures like fingerprints, face, retina, palm, etc. This kind of process can be expensive as the sensors need great accuracy in order to guarantee the correct authentication of the person. The decision of which one to use is based on the security the organization needs, for instance, a retina scanner is very trustworthy, but the prices are exorbitant. On the other hand, fingerprint scanners are now integrated in a lot of devices such as mobile phones, smartwatches and laptops which gives them accessibility for more users.

These identifiers are based on pattern recognition, that is being achieved via image comparations. For instance, fingerprints have patterns like arches, whorls or loops that make each fingerprint almost unique. These detailed forms are called *minutiæ* and define the points to be stored for later comparation each time the user tries to authenticate. But this pattern recognition is not perfect as the image match score is not 100% every time, for example, if the user moves the finger one millimeter. Thus, the system defines an **acceptance threshold** which defines the tolerance of the match score. Now, if the real user and fake users tried to authenticate, the following graphic would be obtained:

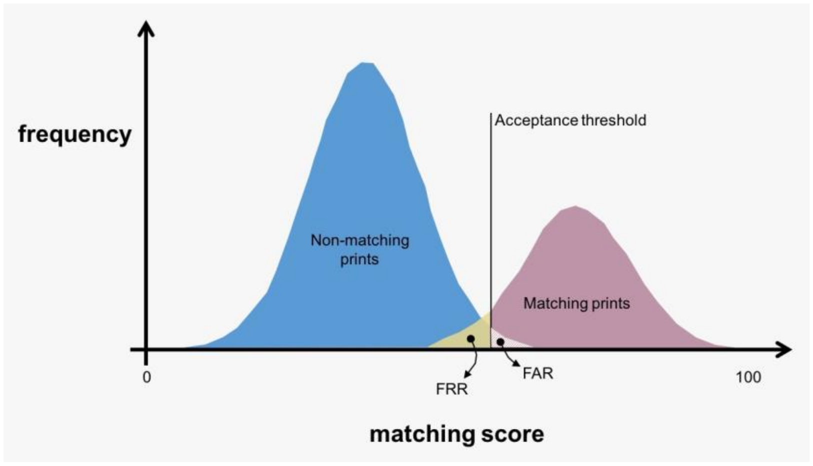


Fig. 1. User's acceptance distribution with fingerprints [7]

In the figure, the purple zone represents the matching score of multiple attempts of the valid user, the blue zone is the same but for an invalid user. The two zones follow a normal distribution and there is an intersection area between them, that is divided in the FRR (False Reject Rate) and the FAR (False Accept Rate). As their names indicate, the first, includes all the users accepted, that are not supposed to get into the system. On the other hand, the second are all the users that are rejected and were legitim. This threshold must be adjusted depending on the reliability the system needs, considering that increasing the threshold would eventually eliminate the FAR with the cost of usability as more legitim users could require multiple tries to authenticate themselves. [7]

Physical authentication mechanisms vary from OTPs (one-time password generators) or hardware tokens and registered devices as a mobile phone, smartwatch, earphones, etc. This kind of mechanism trust in the fact that the only user that has access to those devices is the legitim user, and it is widely use in environments that need high security like banks. Lately, more webpages are adapting this kind of authentication as a second factor used to improve the user's security.

Finally, the user's information that do not imply memorizing something are based on identifiers that use the user patterns. These patterns are analyzed in order to determine the authenticity of the user and discarding all the attempts that do not match the given patterns. Some examples of used patterns are geographic location, network identifiers (addresses, devices, etc.) and behavioral (hours, gestures, routines, etc.).

In order to determine the correct authentication method, the advantages and disadvantages in the specific context must be analyzed. First, the unique identifiers guarantee the authenticity of a person very accurate and lately this authentication method has become a trend among the people, despite it would imply an inversion in the devices that would be used to authenticate each user every time they want to access a service, along with the acceptance threshold that could cause discomfort among the users in the most secure systems. Second, the physical authentication mechanisms could be a good option, each member could use their own token or register a device they own. However, require the use of tokens would require an increase in the budget that passwords did not have. At last, the methods based on user's information can be secure but depend a lot on the formation managed by the idP. This implies that a method to recollect, clean and analyze the information must be defined, furthermore, an inversion on the required infrastructure for the process.

Consequently, considering that a university can hold up thousands of students the methods based on user's information would require a lot of computational power for each time a user is required to log in. Also, there are some personal that do not follow regular patterns as part of their job, for instance, researchers could need to travel to investigate somewhere as part of their investigation, or teachers that receive congress invitations on other countries. Thus, this method must be discarded as it implies more inversion on infrastructure, human resources that manage the new service, and it would be too inconvenient for some of the users that require the authentication service.

Even though, the biometrics and physical authentication devices represent a new inversion as the services need to be changed and some devices will be bought, both do not require the hiring of new specialized personal. Also, the devices do not require constant supervision or regular maintenance as the new servers for the last method require. On the other hand, some expenses like the customer service for the password retrieval or account retrieval after a lot of attempts would be dismissed, along with all the attacks related with passwords. Therefore, these two methods will be considered as the main objective of the project.

Now, the passwordless environments are not something "new" on the market, the first solutions were deployed in 2009 while trying to use biometrics for online payments. Later, an organization called the FIDO alliance (Fast Identity Online) started to develop some standards with the intention of get to know this kind of solutions and make them available for any organization that wanted to use it. First, the FIDO Universal authentication framework (UAF) was released alongside with FIDO universal second factor (U2F), this became the standards for web passwordless-based solutions and were implemented by enterprises like Microsoft, Facebook, google, etc. After that, with the collaboration of the World Wide Web Consortium (W3C), they developed the standard FIDO 2.0. [8] The idea behind the protocols, and the objective of the FIDO alliance, is to reduce the usage of passwords and facilitate the authentication process for the final user, as the methods proposed are based on simplicity and the user's trust on them.

First, the U2F protocols allows the user to enable a second factor authentication method with the objective of increase the security of the account at the login stage. Some of the methods supported are external tokens, NFC compatible devices like smartphones, biometrics, phone text, etc. The process starts with the user attempt to login, where the user/email and password must be provided, after checking the password, the second factor is requested as registered by the user. With this, the passwords can be simplified as the security relies on the second factor.

On the other hand, the UAF protocols allows the user to authenticate itself using passwordless or multifactor authentication. The process starts with the registration of the authentication method, which are the same as U2F, after that, each time the user wants to login, the password will not be needed, as all the authentication is based on passwordless. Furthermore, the user can request to use more than one method, for instance, fingerprint + face recognition, or any combination they choose.

Finally, FIDO 2 is a set of specifications that join the Client-to-Authenticator Protocols (CTAP) from the FIDO alliance and the Web Authentication (WebAuthn) specification of W3C. This standard supports the previously mentioned methods (passwordless, second factor and multi-factor) with an extended range of authenticators like wearable devices. Since the later manages the selected authentication methods, it will be used on the solution design and deployment. However, the basis of this specifications must be understood first.

3.1 FIDO 2

A passwordless environment is based on the public-key cryptography, which is a method where two different keys cipher the information instead of the classic model of symmetric cryptography where the message had to be ciphered and decrypted with the same secret key. Those keys are known as the private-public key pair and are widely used on web protocols for private communication between two parts, private key agreements and the digital signature of the websites; this means that they provide confidentiality and no repudiation.

The system is based on the following statement: the public key is known by everybody, and each of them can use it in order to encrypt a message for the owner of the key. Once the message is received, the owner can decrypt it using the private key that **no one else** knows and obtain the message. This implies that there is confidentiality between the two sides as if someone catches the ciphered message, it cannot be understood by any mean, unless they possess the private key. However, this does not work both ways, as when the entity ciphers a message with the private key, the message can only be deciphered with the public key, which, anybody could know. On the other hand, this process provides authenticity as only the owner of the private key could have encrypted the message, thus, if a given message can be decrypted with the public key associated to one entity, it is assured that the message comes from that entity. The next graphic provided by *Sectigo*, shows a brief of the described process:

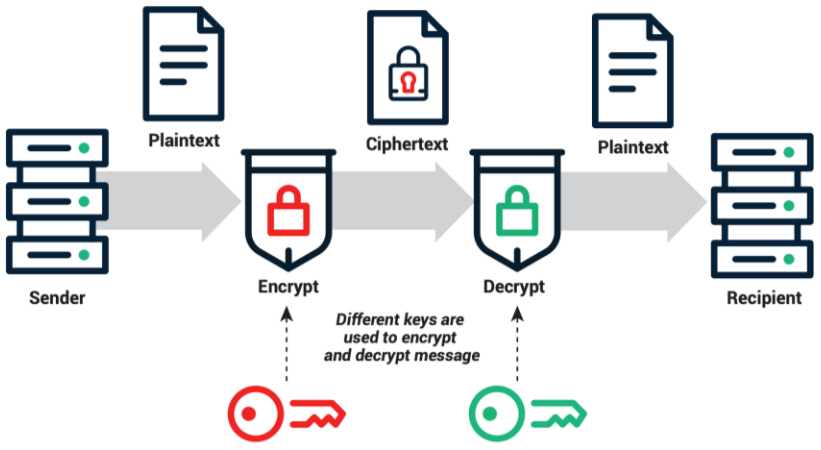


Fig. 2. Public-key cryptography process [9]

Using this concept, if a person is given a private key to guard and use, they can authenticate themselves as long as they keep the key secret. Nevertheless, this is different of a password as it does not depend on the person knowing something to confirm their identity, instead, they possess something that is only theirs. With that in mind, a lot of devices can be used to achieve this, like the private seal of the king on the ancient times, but with more daily devices as a smartphone, smartwatch or a token.

Additionally, to fully understand the process exposed by FIDO 2 specification three main sections must be covered. First the initial passwordless protocols, U2F and UAF, then the WebAuthn API and lastly the CTAP protocol. The first protocols are included here as FIDO 2 supports them in the authentication process. However, the core of FIDO 2 is composed by the WebAuthn API and the CTAP, these two specifications provide the complete panorama for a secure and user friendly passwordless environment that covers server, client and Authenticator communication.

3.1.1 FIDO 1

The FIDO standards were launched with the purpose of creating a stronger authentication process that helped to mitigate the damage made by passwords, decreasing data breaches and identity fraud. Also, the program wanted to make an easier, faster and convenient login process for the users as they do not have to remember and type passwords, and in most cases, buy new devices [10]. In this section both standards will be explained.

In this case, the standards manage a series of "Authenticators" which refer to the devices that manage the keys for the protocol. As stated along the paper, these devices include cellphones, wearable devices, USB tokens, biometrics, among others. Likewise, they are divided in two categories, on-device authenticators like biometrics or in some cases cellphones, and external authenticators which include the USB tokens, cellphones, wearable devices, or NFC/Bluetooth tokens.

- **UAF**

  This set of protocols manages passwordless and multi factor authentication security, which includes methods like biometrics, voice recognition, pins or registering a device for authentication. All these options are presented to the user who must select one or more of them through the registration process. Then, the user can authenticate itself to a service provider by repeating the selected authentication method(s).

  The high-level architecture of this specification is the following as presented on the normative FIDO specifications overview [11]:

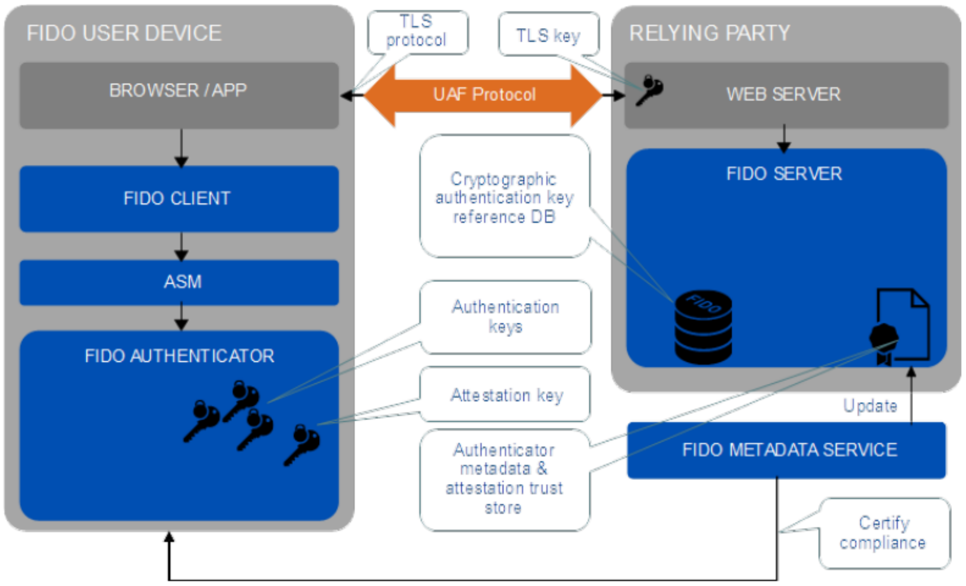


Fig. 3. UAF High-Level Architecture [12]

As shown in the image, the architecture is composed by three main actors: User's device, RP server and FIDO metadata service, however, the protocol is defined between the UAF client (user's device) and the UAF server (RP server). Each one has a sub structure according to their functionalities and usage scenarios, and the model does not provide information regarding deploy environments or devices as it was designed to work across different platforms. Moreover, the FIDO related components are colored with blue and the 3rd party applications are colored in gray.

First, the user's device is composed by the user agent, the FIDO UAF client and the Authenticator. The first, is the interface that interacts with the user along the authentication process, most of the times a web browser or an application. This is the responsible of stablishing a secure channel via TLS to communicate with the RP, where the protocol will be executed. Secondly, the UAF client oversees the communication between the user agent and the selected authenticator (external or on-device). The interaction with the Authenticator is made using the FIDO UAF Authenticator API.

On the other hand, the RP is composed by the web server and the FIDO UAF Server. Here the web server manages the communication with the user agent via TLS. The UAF server has three main functions. First it must validate that the user's Authenticator device is part of the trusted registered Authenticators, this is reached via metadata exchange where the devices that are not found in the registered Authenticators cannot finish the registration process. Secondly, it manages the association between the registered devices and the users at the registration. Finally, it verifies the authentication and validates the process.

Authenticators

The Authenticators are known as "a secure entity" and must be registered as certified by fido, otherwise the registration process will fail. Those devices are capable of create keys that will be associated to a RP during the registration and authentication process, they must use a strong PRNG and a good source of entropy. Also, they must attest their type (e.g., biometric), capabilities (e.g., supported crypto algorithms) and their provenance. The final authentication is reached by a cryptographic challenge using the registered keys.

Metadata validation

The metadata is used at the registration process in order to verify the authenticity of the Authenticator. This message is called “Attestation” during the process and it is signed with the private key generated for the specific service. On the other hand, the public key is found in the Authenticator metadata and it is shared with the UAF server out of band.

UAF Protocols and use cases

The UAF standard supports five protocols: Authenticator registration, user authentication, set-up authentication, secure transaction and authenticator deregistration. However, only the relevant for this project will be delve in. The following are their capabilities and processes:

- *Authenticator registration:* It can discover the available authenticators on the user chosen device, verify the attestation assertions made by the Authenticators (verify that they are certified and trusted) and register the device.
  The process starts from the user agent when it initiates the registration process, there, the browser must identify if an Authenticator is available, if not, the process cannot continue. Once the UAF server receives the initial message, it sends a registration request and its policy to the UAF client, with that, the Authenticator enrolls the user and generates a **new key pair** associated to the specific RP so the UAF client can finish the registration process by sending the registration response, attestation and the user’s public key. Finally, the UAF server validates the received message as explained in “Metadata validation” and if everything is fine, the public key is stored and associated to the user for future interactions.
  The following image provided by the FIDO Alliance illustrates the protocol:

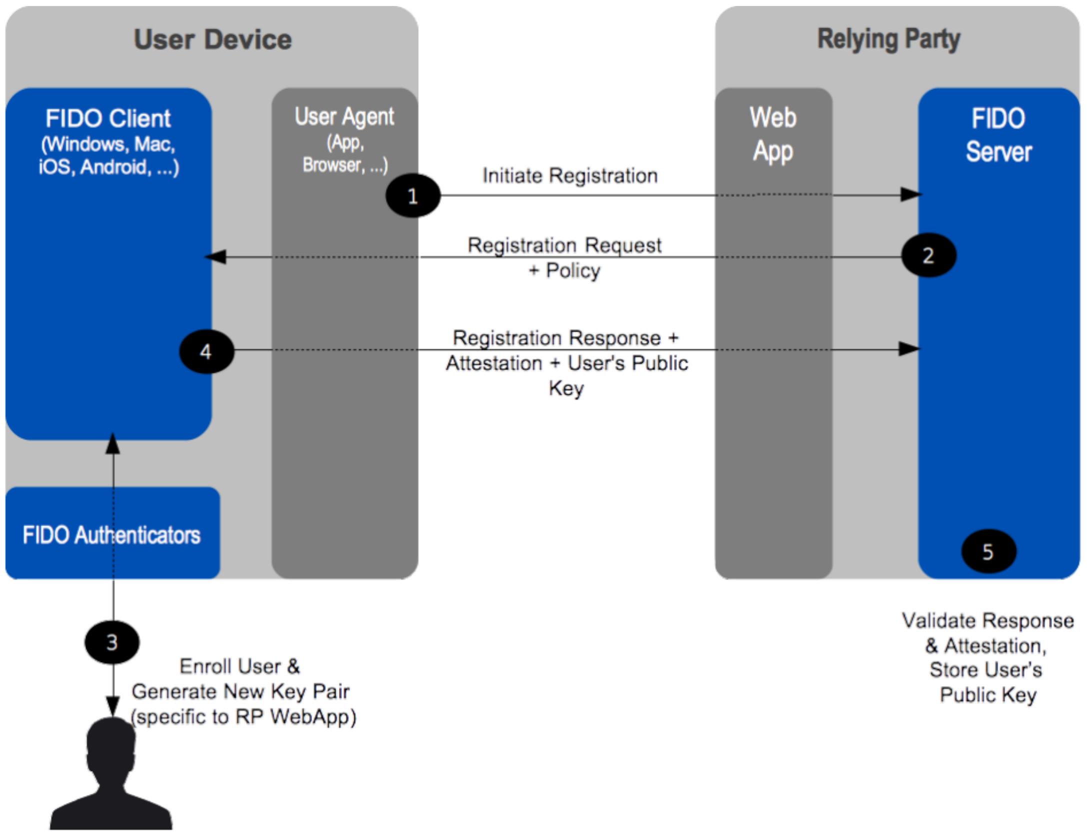

User Device
Relying Party
FIDO Client
(Windows, Mac, iOS, Android, ...)
User Agent
(App, Browser, ...)
Web App
FIDO Server
1
Initiate Registration
2
Registration Request + Policy
4
Registration Response + Attestation + User's Public Key
FIDO Authenticators
5
3
Enroll User & Generate New Key Pair (specific to RP WebApp)
Validate Response & Attestation, Store User's Public Key

Fig. 4. UAF registration message flow [13]

- *User authentication:* The authentication is reached via cryptographic challenge-response protocols. They are independent of the authenticator that is used, but the user must have registered before and use the same device used in the registration process. If the user does not count with that authentication method at the time, it is responsibility of the RP to provide an alternative login or inform the user that it is not allowed to login.
  The process starts from the user agent when the user wants to login, there, the user agent sends the initiate Authentication message to the RP. The RP proceeds to send the Authentication request, the policy and the challenge signed with the public key registered in the authentication process. When the UAF client receives the message, it sends the challenge to the Authenticator, which must first verify the user, then use the private key associated to the wanted service and respond the cryptographic challenge. Finally, the UAF client sends the authentication response signed with the private key.
  The following image provided by the FIDO Alliance illustrates the protocol:

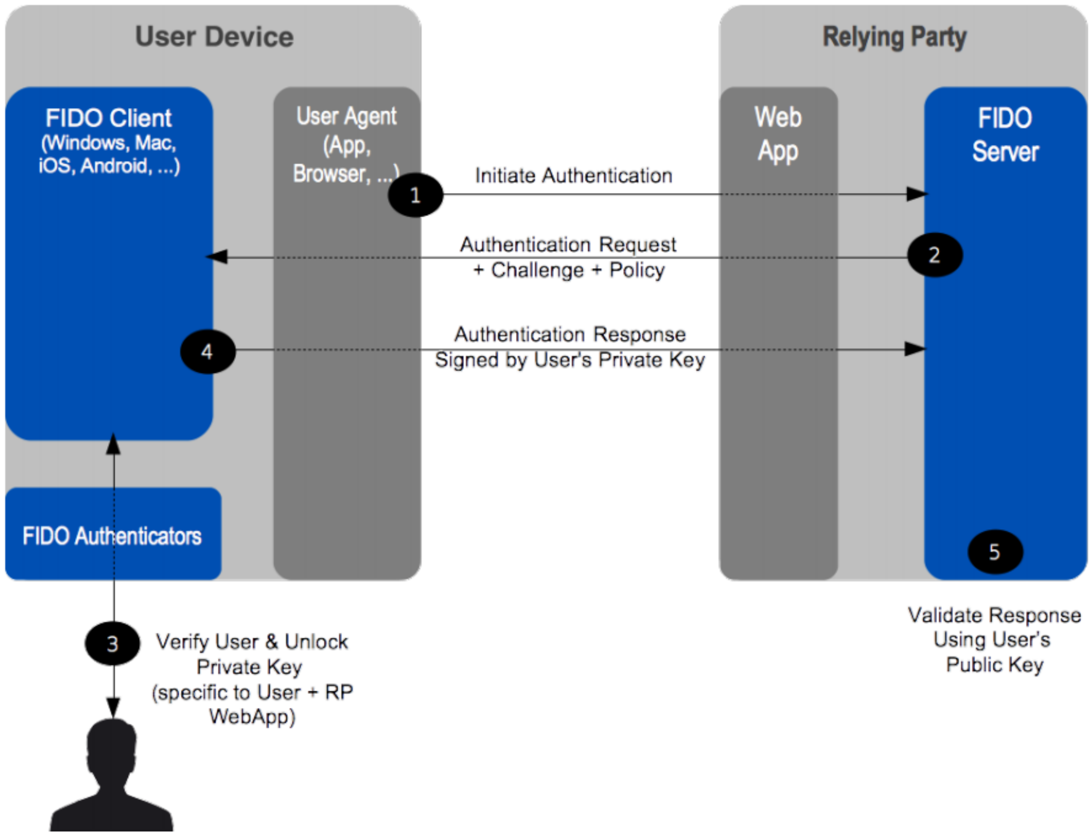


Fig. 5. UAF authentication message flow [14]

- *Secure transaction confirmation:* Is used to confirm a transaction and it is defined by the RP. It can only be done if the Authenticator has the capability of doing it.
- *Set-up Authentication:* This protocol can be used by the RP whenever they need or have a service that requires higher-assurance authentication. For instance, the user authenticates itself like any other user, but certain actions will require the extra authentication using the UAF devices. The initial login is not required to be with UAF.
- *Authenticator deregistration:* Deletes the associated user's UAF credential of the RP and the Authenticator.
  Whenever the user needs to deregister an Authenticator (for example, the token was lost, or the user is leaving the company) it must contact the RP who will delete the associated public key in the UAF server and send a Deregistration Request to the UAF client. Then, if

available, the client informs the Authenticator and the private key associated to the service is deleted.

The following image provided by the FIDO Alliance illustrates the protocol:

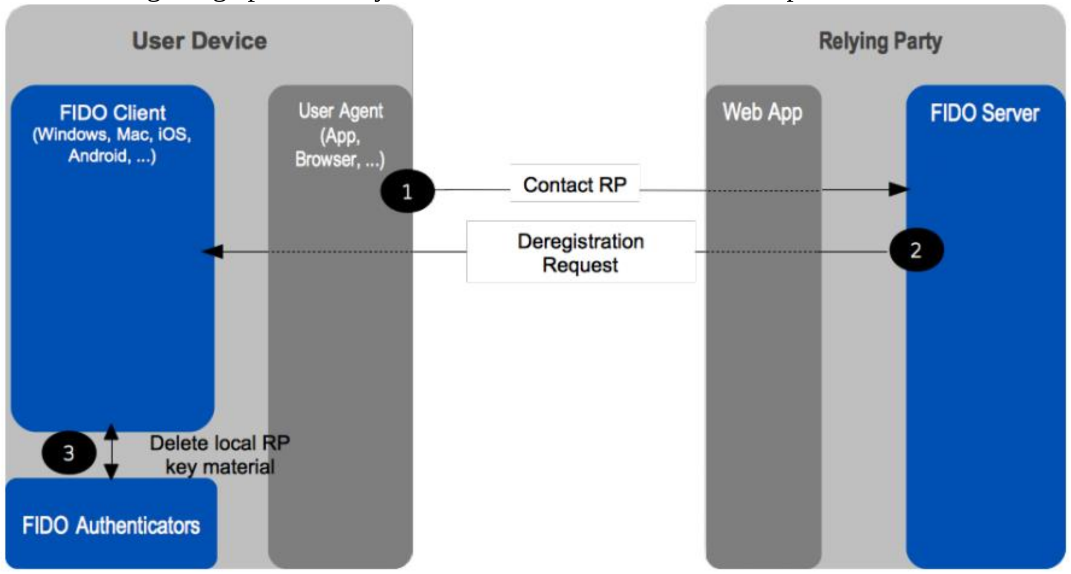

User Device
Relying Party
FIDO Client (Windows, Mac, iOS, Android, ...)
User Agent (App, Browser, ...)
Web App
FIDO Server
1
Contact RP
Deregistration Request
2
3
Delete local RP key material
FIDO Authenticators

Fig. 6. UAF deregistration message flow [15]

- **U2F**

This set of protocols manages a strong second factor authentication over the classic password-oriented environments. The objective is to substitute the password for easier passwords as the security relies on the second factor with options like PIN + biometric, thus, the security is not compromised, and the user keeps login as usual.

The authentication methods are also the Authenticators. However, the protocol changes as the second factor considers another factor that is the “user presence” when doing the login. Moreover, the cryptographic strategies that are core of the protocol remain and some considerations in the communication with the Authenticators are presented. All of this is documented in the FIDO alliance documentation.

The architecture of this specification is the following as presented in the FIDO Universal 2nd Factor (U2F) Overview specifies [16]:

The specification is divided in two sections, the JavaScript API and the Authenticators protocols (NFC, Bluetooth and USB). Thus, the protocol is designed to be used **only on web environments** as it relies on the browser JavaScript execution for the communication protocol between the client and the web server (in UAF the protocol was hybrid and could be used for web and mobile applications). The following figure gives an example of a possible architecture for the U2F specification.

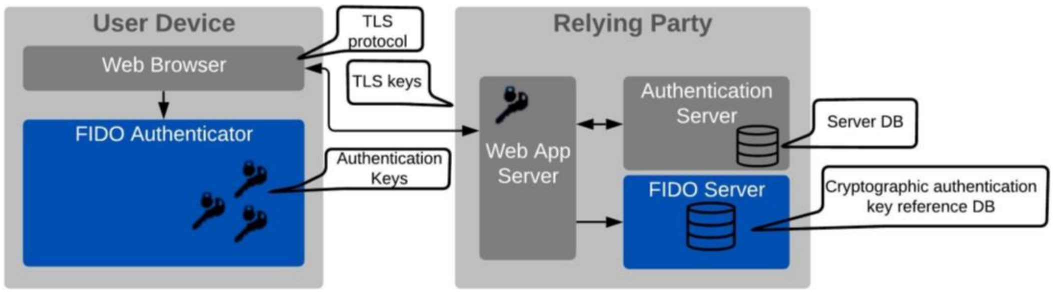


Fig. 7. U2F High-level architecture[1]

As the image shows, the Authenticators for U2F are not necessarily certified by FIDO Alliance, therefore, the RP must check that the Authenticators are good enough for their requirements (checking, for instance, the cipher algorithms it supports or the chosen PRNG). Some additional metrics are given in the FIDO U2F Implementation considerations standard. Moreover, the standard does not define a protocol for deregistration, therefore, the RP must define ways to solve this problem. Some of the solutions is defining an alert when something happens to the token and invalidating and deleting the keys stored in the FIDO server.

Moving on, the protocol presents two use cases Registration and Authentication, were both use the protocol related to the Authenticators and the JavaScript API. However, the protocols are not fully described as the initial authentication process is highly connected to the actual state of each authentication server, as it is more like an extension to the password-based authentication process. Therefore, the next models were generated as an example of the protocol:

- *Registration:* When the administrator allows U2F, on the next successful login the user will be requested to present the token for registration. In this process, the browser informs the Authenticator, which must respond the request creating a key pair after its activation (the activation is when the user confirms their presence by clicking a button or similar, preventing malware to use the tokens when the user is not active), that will be send to the web browser, who will redirect the response to the FIDO server. Once the FIDO server receives the answer, it associates the key with the service and the user for later use in the authentication process. The next figure is an example of the process followed for the registration.

[1] This section graphics do not represent FIDO Alliance normative.

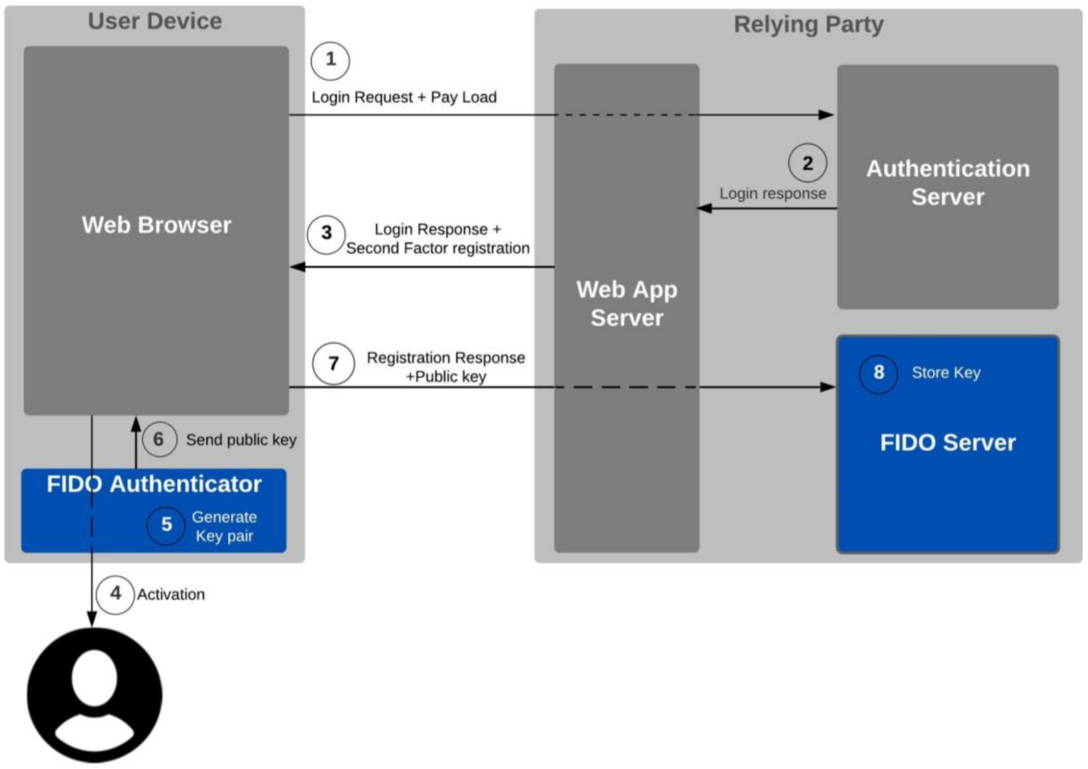


Fig. 8. U2F registration message flow

- *Authentication:* The user starts the password-based authentication process as usual. Then, the FIDO server requires the second factor authentication, that the user must confirm in order to login. However, as the browser does not know which Authenticator contains the key that the service requires, therefore, it sends a packet called "client data" with the signature response of all the U2F devices that answered to the RP. After the RP receives the message, it verifies if any of the signatures matches the expected.
  Essentially, the client data not necessarily comes from multiple devices. If one U2F token posses' multiple keys, it will answer with the signature of each one of those keys as it does not necessarily know which of them is associated to which service.

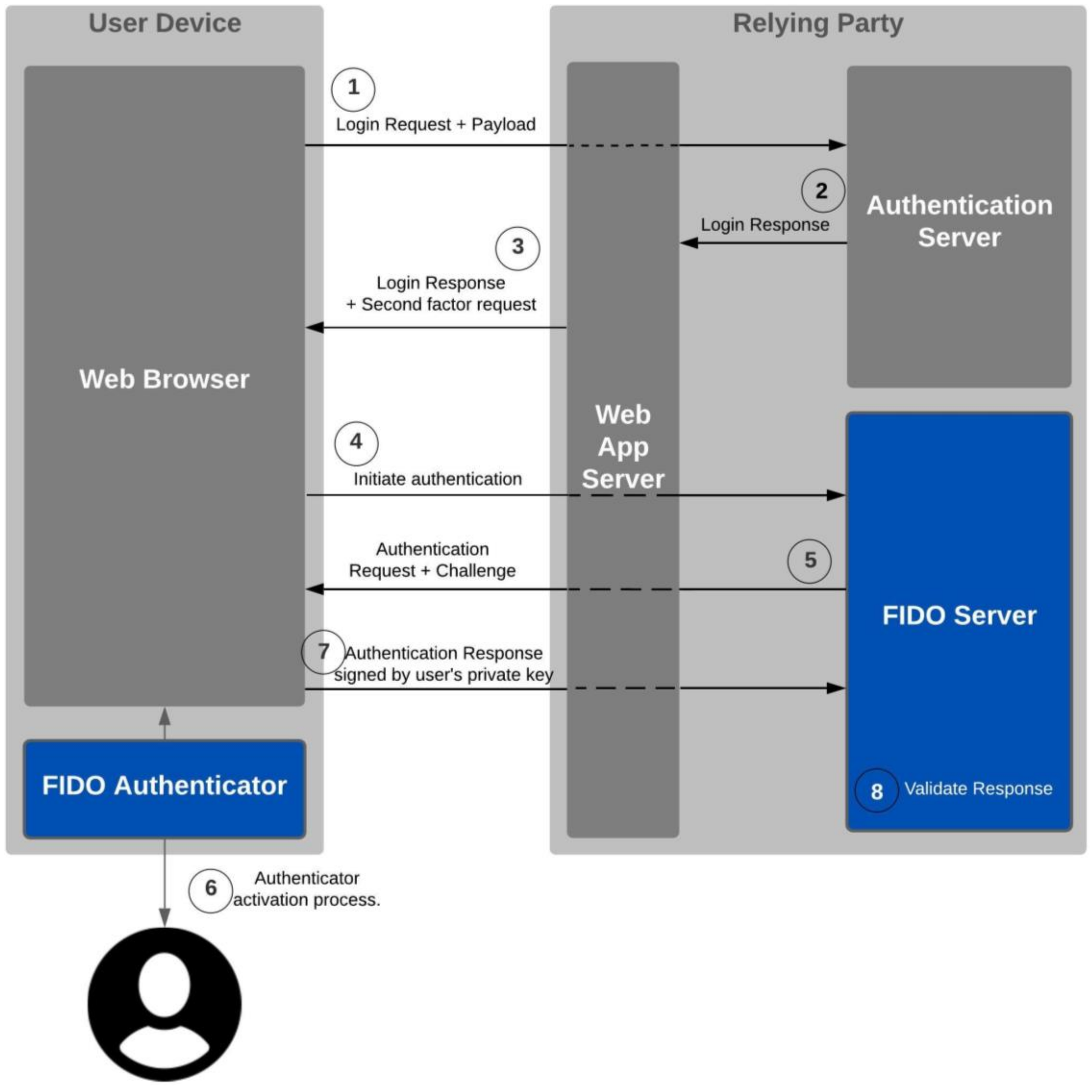

User Device
Relying Party
Web Browser
FIDO Authenticator
Web App Server
Authentication Server
FIDO Server
1 Login Request + Payload
2 Login Response
3 Login Response + Second factor request
4 Initiate authentication
5 Authentication Request + Challenge
6 Authenticator activation process.
7 Authentication Response signed by user's private key
8 Validate Response

Fig. 9. U2F authentication message flow

3.1.2 WebAuthn

"This specification defines an API enabling the creation and use of strong, attested, scoped, public-key based credentials by web applications, for the purpose of strongly authenticating users." [17] For this purpose, it defined two protocols registration and authentication that will work between the three defined conformance classes: User agents, Authenticators and WebAuthn Relying Parties (RP). The following is a description of the specification as given by the official W3C documentation [17]:

Actors

The API defines three conformance classes User agents, Authenticators and WebAuthn Relying Parties. In order to be considered conformant, each one of this **must** behave as it is indicated in the WebAuthn standard.

- User agents:

It is the intermediary between the Authenticators and the RP, located in the client device. Distinct to the other two conformances, this one can vary the algorithms as they please, with the condition that the results follow the standard and are “indistinguishable from the result that would be obtained by the specification’s algorithms.” [17] Additionally, most of the protocol is managed through this conformance, thus **it mediates all the operations in the protocol** redirecting the protocol messages.

On the other hand, it is the part of the application that communicates directly with the user, hence, the software that supports this conformance should have a user interface that allows it to manage all the require operations for the use cases.

Besides, it manages the largest security layer (alongside to the Authenticator) as it must prevent RP to access unowned credentials and guarantee that the communication channels are secure. The first can be achieved as it can check the origin of each message in the protocol and confirm the validity of it according to the RP identifier.

- Authenticators:

Refers to the cryptographic entity used by the WebAuthn client to fulfill the following tasks:

  - Generate a key pair and associate it to a RP.
  - Provide authentication by verifying the user.
  - Respond the cryptographic challenge presented during the authentication process in an *Authentication Assertion.*

This entity can be hosted in different *platform authenticators* like cellphones, computers, wearable devices, etc. or in *roaming authenticators* like USB tokens. Also, they only interact with the web browser and in the case of biometrics, that information never leaves the device. Moreover, the standard defines authenticator types according to their “attachment modality, employed transport(s), credential storage modality, and authentication factor capability.” This will be used to determine the use cases that each type supports.

Finally, an Authenticator must perform the following operations: **lookup Credential Source by Credential ID, create credentials, get Assertion,** and **cancel one of the previous process at any moment**.

- WebAuthn Relying Parties:

This conformant is the web application interested in the authentication service using the API. It must take careful about not leaking information during the protocol and it must manage the **registration of a new credential** and **verifying an Authentication Assertion**.

Attestation

The attestation messages are how a RP verify the authenticity of an Authenticator. The capacity to generate this kind of message is an obligation of an Authenticator and must use it during the protocol.

It is composed by the signature generated with the private key pair of the registered public key saved on the WebAuthn RP, the challenge sent by the RP or the response to it, and a certificate or similar (i.e., metadata) with the information of the public key. The last can also contain additional information as the manufacturer, a certifying agency or similar. With this information, the RP is capable of trust the Authenticator during the authentication process.

Use cases

As stated in the Authenticators section, there are different kinds of authenticators, consequently, the use cases have different scenarios. The possible scenarios are phone registration, computer without *platform authenticator* registration, computer with *platform authenticator* registration, and solo or hybrid authentication (referring to the number of devices used).

- Registration

First the user must login into the domain they want to register using the actual authentication method they have, or the user can create a new account. After this depending on the device and its capabilities the process continues as follows:

On a phone or a computer with *platform authenticator* a notification/prompt will appear asking the user if it wants to register the device on the domain. If the user agrees, the device will ask the user to use a previously configured *authorization gesture* (PIN, biometric, etc.). And the registration is completed, a message must be shown.

On a computer without *platform authenticator* the user must navigate to the settings and search for the option "Register security key", the user then must connect the Authenticator that will be used (plug in the USB token or pair the Bluetooth/NFC Authenticator). Then the web page indicates the user must confirm its presence by clicking a button or approaching near the NFC device. The registration is completed, a message must be shown.

- Authentication

In the case of **hybrid authentication,** the user must pair their phone or NFC device or plug in the USB token before starting the process. After that, the process for both options starts with the user accessing the domain of interest and initiating singing in.

For hybrid authentication: if the Authenticator is a paired phone, a notification will prompt asking the user to confirm their identity (if there are multiple accounts registered on the device), then, the *authorization gesture* prompts, the user completes the process and returns to the device where he/she is login in.

If the Authenticator is an NFC or USB token the user must provide the *authorization gesture* (getting the NFC device near or pressing the button in the token), then the user completes the process and returns to the device where he/she is login in.

For solo authentication: The user provides its username and the device's browser will show a prompt asking to complete the previously configured *authorization gesture.* The user completes the action.

After this, the web browser shows the service with the user signed in.

Registration protocol

The user accesses the web service and if not already, logs in using the method accepted by the RP. If the user does not have an account, it can be created, and the process continues. The RP sends a message solicitating the available Authenticators to the client platform, which search and locates them. Then, the client platform stablishes a connection with the authenticator(s) (the connection may vary from device to device) and if necessary, ask the user to select the wanted Authenticator. The selected Authenticator will ask for the configured *authorization gesture* before creating the new credential and the attestation, and send it to the client platform, which redirects the response to the RP if the process was completed successfully, otherwise an error message is shown to the user and the process must start again.

When the RP receives a new credential, it sends the generated public key and the attestation to the server which confirms that the Authenticator fulfills the requirements of the conformance section using the attestation message. If it does, the server associates the public key to the user in their database and stores it, optionally, the server can store the authenticator type in order to provide a more fluid user experience.

As a matter of fact, the protocol supports legacy second factor authentication with FIDO U2F. For this, the RP must accept the attestation empty as FIDO U2F devices do not provide metadata about their manufacturer or certificates. However, the decision is taken by the RP as it is not as secure as the certificated tokens. Moreover, this protocol does not provide a standard for communication with the authenticators but is important that the RP manages secure connections between itself and the Authenticators.

The next figure is an example of the registration protocol:

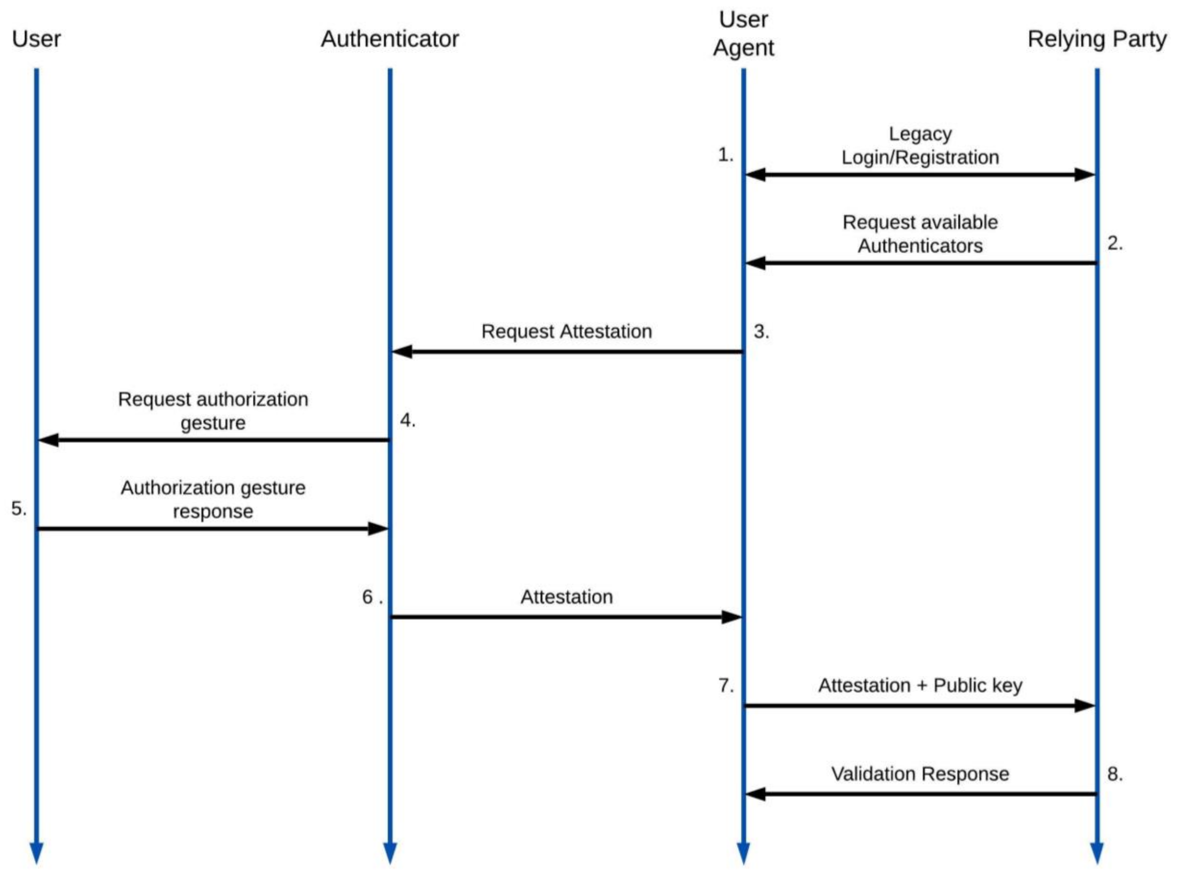

User
Authenticator
User Agent
Relying Party
1. Legacy Login/Registration
2. Request available Authenticators
3. Request Attestation
4. Request authorization gesture
5. Authorization gesture response
6 . Attestation
7. Attestation + Public key
8. Validation Response

Fig. 10. WebAuthn registration protocol

Authentication protocol

The user accesses the web service and starts the authentication process. The web browser asks the client for an *Authentication Assertion* that allows it to authenticate itself, this can be achieved by providing a username or using the Authenticators data (optional). After this, the RP creates a cryptographic challenge using the previously registered public key credential of the user and sends it to the client. The client must identify and connect with the Authenticator, send it the received challenge and wait for the answer. The Authenticator then shows the information regarding the origin of the authentication request to the user, asks for a confirmation and then it ask for the *Authorization gesture*. With this, the Authenticator responds the challenge and returns the information to the client who redirects this information to the RP if it satisfies the configured requirements.

When the server receives the *assertion* generated by the Authenticator it extracts the information and searches the public key credential in its database. With this, it proceeds to verify the *assertion's authentication signature*. With this the protocol finishes as the next steps change depending to the requirements of the RP. This means, the RP must manage the success or failure cases in the authentication process informing the user about the end of the process.

The next figure is an example of the authentication protocol:

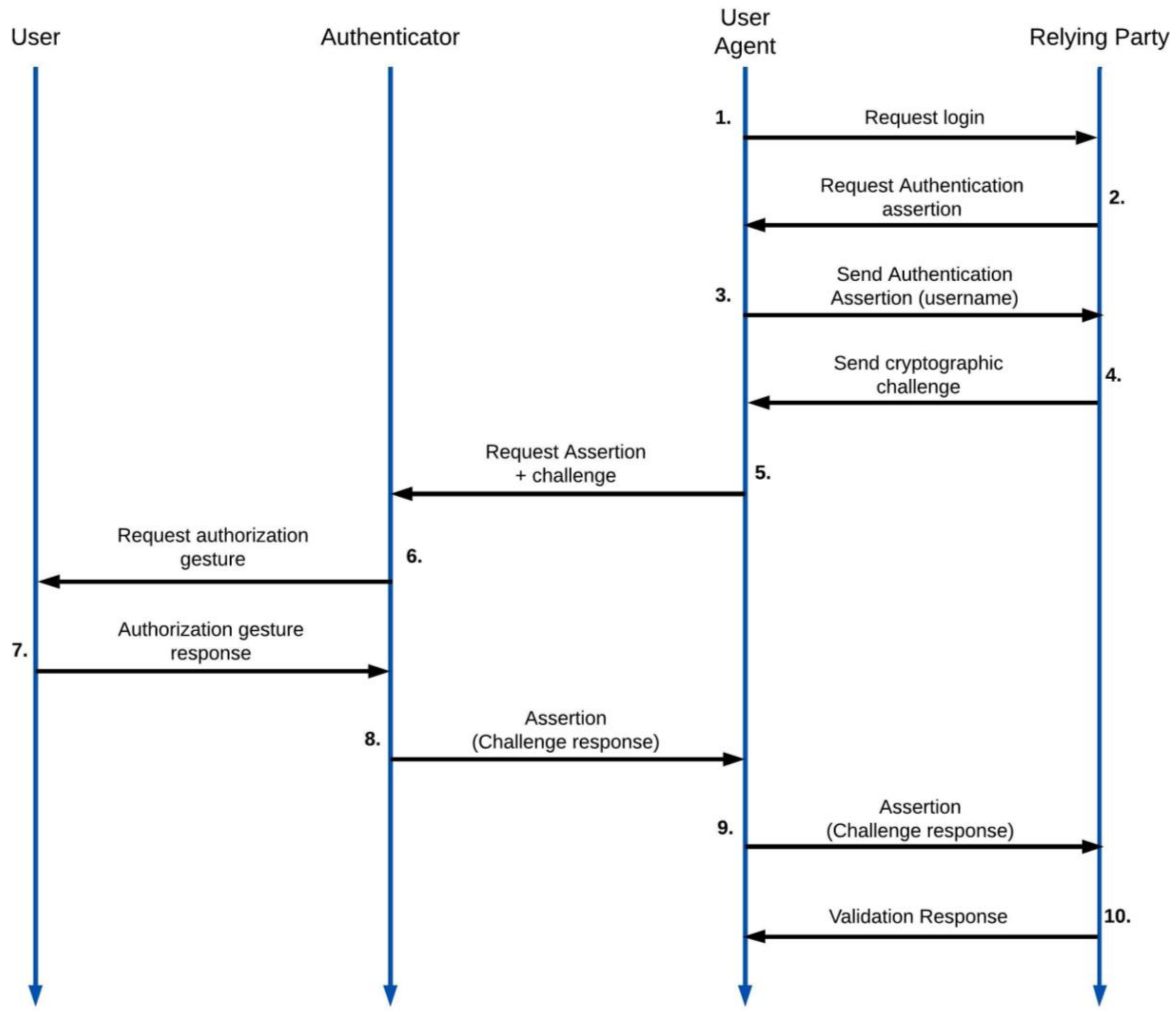

User
Authenticator
User Agent
Relying Party
1. Request login
2. Request Authentication assertion
3. Send Authentication Assertion (username)
4. Send cryptographic challenge
5. Request Assertion + challenge
6. Request authorization gesture
7. Authorization gesture response
8. Assertion (Challenge response)
9. Assertion (Challenge response)
10. Validation Response

Fig. 11. WebAuthn authentication protocol

3.1.3 CTAP

The CTAP specification presents a standard for the communication between the Authenticators and the client (or user agent in WebAuthn), and the specific bindings required to connect the client with the different types of Authenticators available on the market. For this purpose, a general protocol for the Authenticators and the client, and a protocol for binding with each one of the supported devices were defined. This document intends to present an overview of these protocols as stated on the Client to Authenticator protocol (CTAP) 2.0 standard document [18], and later the design section explains the used protocols on detail.

First, the CTAP protocol has two versions CTAP1/U2F protocol and CTAP2 protocol. The first one was developed and launched to the public to continue with the usage of the U2F protocols and devices, as the UAF and FIDO2 protocols present new security measures regarding the previously mention Authenticator metadata validation (refer to UAF and WebAuthn Authenticators section) that the U2F tokens do not possess. On the other hand, the CTAP2 protocol implements the new security measures and allows that the devices that are compatible with U2F, continue working mapping the CTAP2 requests to CTAP1/U2F requests and the CTAP1/U2F responses to CTAP2 responses. Moreover, it introduces new devices to the available Authenticators defining the bindings for Bluetooth, NFC and the new certificated tokens.

The mapping between these two protocols is needed mostly for the attestation and the credential creation processes because the U2F tokens do not possess the metadata information necessary for the attestation process and later Authenticator registration to the server. Consequently, the standard presents the necessary modifications to the RP for the validation of the metadata and registration of the new token. Even though they are present, it is not necessary to implement this as the platform may choose to work only with CTAP devices.

Regarding the general protocol, it is defined as these four steps:

1. Platform stablishes the connection with the authenticator.
2. Platform gets information about the authenticator using *authenticatorGetInfo* command, which helps it to determine the capabilities of the authenticator.
3. Platform sends a command for an operation if the authenticator is capable of supporting it.
4. Authenticator replies with response data or error. [18]

Where the platform refers to the web browser of the user. Additionally, as it is seen in the point three, the CTAP1 protocol is considered in this general protocol. However, the main point of the protocol is not in these four steps, it is in the **protocol structure** which has three parts the Authenticator API, the message encoding and the transport-specific Binding.

3.1.3.1 Authenticator API

During the protocol, the Authenticator is treated like an API that the platform must implement. This means that the Authenticator has default functions that the client (user agent) will call constantly when interacting with it. These functions receive some predetermined parameters and must return an output or an error code when needed. Moreover, despite all the operations are independent and asynchronous allowing the Authenticator to attend various requests, a limit according to the device capacity must be set to prevent possible failures or damage to the device.

The API must contain the following methods:

```
authenticatorMakeCredential(clientDataHash, rp, user, pubKeyCredParams, excludeList,
extensions, options, pinAuth, pinProtocol)

authenticationGetAssertion(rpId, clientDataHash, allowList, extensions, options, pinAuth,
pinProtocol)

authenticatorGetNextAssertion()

authenticatorGetInfo()

authenticatorClientPIN(pinProtocol, subcommand, keyAgreement, pinAuth, newPinEnc,
pinHashEnc)

authenticatorReset()
```

The FIDO authenticators vendors are in charge of implementing this inside all of their Authenticators and register the metadata information when the certificate is provided by the FIDO Alliance. Also, they should verify their devices security measures and if any vulnerability is discovered, update the devices or inform that they are no longer safe to use in the metadata.

3.1.3.2 Message Encoding

As the communication channels of some of the Authenticator devices have a bandwidth-constraint (e.g., Bluetooth) the specification defined a light encoding that allows to effectively transmit the information between the authenticator and the client platform. This encoding uses the concise binary encoding CBOR and with the objective of "reducing the complexity of the messages and the resources required to parse and validate them" [19] all the messages, encoders and decoders must use the called **CTAP2 canonical CBOR encoding form**. That follows the next rules:

- Integers must be encoded as small as possible.
  - 0 to 23 and -1 to -24 must be expressed in the same byte as the major type;
  - 24 to 255 and -25 to -256 must be expressed only with an additional uint8_t;
  - 256 to 65535 and -257 to -65536 must be expressed only with an additional uint16_t;
  - 65536 to 4294967295 and -65537 to -4294967296 must be expressed only with an additional uint32_t.
- The representations of any floating-point values are not changed.
- The expression of lengths in major types 2 through 5 must be as short as possible. The rules for these lengths follow the above rule for integers.
- Indefinite-length items must be made into definite-length items.
- The keys in every map must be sorted lowest value to highest. The sorting rules are:
  - If the major types are different, the one with the lower value in numerical order sorts earlier.
  - If two keys have different lengths, the shorter one sorts earlier;
  - If two keys have the same length, the one with the lower value (byte-wise) lexical order sorts earlier. [19]

Because some authenticators are memory constrained, the depth of nested CBOR structures used by all message encodings is limited to at most four (4) levels of any combination of CBOR maps and/or CBOR arrays. Authenticators MUST support at least 4 levels of CBOR nesting. Clients, platforms, and servers MUST NOT use more than 4 levels of CBOR nesting. [19]

Likewise, because some authenticators are memory constrained, the maximum message size supported by an authenticator MAY be limited. By default, authenticators MUST support messages of at least 1024 bytes. Authenticators MAY declare a different maximum message size supported using the maxMsgSize authenticatorGetInfo result parameter. Clients, platforms, and servers MUST NOT send messages larger than 1024 bytes unless the authenticator's maxMsgSize indicates support for the larger message size. Authenticators MAY return the CTAP2_ERR_REQUEST_TOO_LARGE error if size or memory constraints are exceeded. [19]

Finally, as the encoding requires that all the messages to be sent on numeric encoding, the protocol defined the *commands and replies,* which represent the messages that are exchanged between the user agent and the Authenticator and vice versa. Also, it defines forty-seven status codes that are sent in the *replies* indicating the kind of error that occurred while executing a *command.* The following tables present the structure of each message:

TABLE I
Command structure [20]

Name	Length	Required?	Definition
Command value	1 byte	Required	The value of the command to execute
Command Parameters	Variable	Optional	CBOR encoded set of parameters. Some commands have parameters, while others do not. {(Refer to the Authentication API section)}

TABLE II
Reply structure [21]

Name	Length	Required?	Definition
Status	1 byte	Required	The status of the response. 0x00 means success; all other values are errors.
Response Data	Variable	Optional	CBOR encoded set of values

3.1.3.3 Transport-specific Binding

The CTAP2 protocol defines a specific protocol for each possible external Authenticator device. Where each protocol includes a little rationale, the protocol structure, framing and its commands, alongside the specifics of each platform. The described Authenticators are USB Human interface Device (USB HID), “ISO7816, ISO14443 and Near Field Communication (NFC)” and “Bluetooth Smart / Bluetooth Low Energy Technology”.[2]

Following the recommendations of the WebAuthn Authenticators, all the commands executed on this Authenticators are state-less and independent. Thus, each command can be used without the necessity of using another command.

- **USB HID**

The protocol used for data transport is CTAPHID which uses the standard HID. It was chosen as it follows the general objective of a user-friendly system with characteristics like Driver-less installation on most user agent platforms and multi-application support. Also, it has some technologic advantages as the fixed latency response, low protocol overhead and the possibility to have a scalable method for CTAPHID device discovery.

This section covers the CTAP2 tokens with *authorization gestures* like built-in buttons, fingerprints scanners, etc.

Protocol structure

For the transport protocol three new concepts are introduced. A ***transaction*** “is the highest level of aggregated functionality” [22] composed by a request and response ***messages*** that are “divided into individual fragments, known as ***packets***” [22].

These transactions are part of the highest CTAP protocol layer, and only one can be executed at a time. On the other hand, the packets are the smallest data unit represented by the HID reports.

[2] The U2F devices are not included in this section as the specification is defined by the U2F standard.

Commands

The CTAPHID protocol must have the following commands (with their respective command codes), however, the vendor could implement other commands according to their necessities.

CTAPHID_MSG	(0x03)
CTAPHID_CBOR	(0x10)
CTAPHID_INIT	(0x06)
CTAPHID_PING	(0x01)
CTAPHID_CANCEL	(0x11)
CTAPHID_ERROR	(0x3F)
CTAPHID_KEEPALIVE	(0x3B)
CTAPHID_WINK (optional)	(0x08)
CTAPHID_LOCK (optional)	(0x04)

The commands fulfill functions like sending the protocol messages, device discovery, debug options for developers, etc.

- **ISO7816, ISO14443 and NFC**

As a guidance for this protocol, the specifications are given by the standard ISO7816-4 from 2013. This standard defines the Smart card application protocol data unit (APDU) that is the data unit used by the protocol. [3]

Protocol

1. Client sends an applet selection command.
2. Authenticator replies with success if the applet is present.
3. Client sends a command for an operation.
4. Authenticator replies with response data or error.
5. Return to 3. [23]

Commands

NFCCTAP_MSG	(0x10)
NFCCTAP_GETRESPONSE	(0x11)

This protocol only defines the commands essential for the general protocol usage.

- **Bluetooth Smart / Bluetooth Low Energy Technology**

This protocol defines some requirements for the devices that will stablish communication via Bluetooth. For instance, the pairing communication **must be** encrypted with a long-term link key (LTK) and the devices "using Bluetooth Low Energy Technology SHALL conform to Bluetooth Core Specification 4.0 or later [BTCORE]. Bluetooth SIG specified UUID values SHALL be found on the Assigned Numbers website [BTASSNUM]." [24]

Additionally, as Bluetooth data travels wireless and has a long range the platform must consider that the connections **might not** be private as this kind of data transfer media is weak to monitoring and injection attacks.

Protocol

[3] A Smart card is a device as a credit card or a cellphone that can use NFC technology.

1. Authenticator advertises the FIDO Service.
2. Client scans for authenticator advertising the FIDO service.
3. Client performs characteristic discovery on the authenticator.
4. If not already paired, the client and authenticator shall perform BLE pairing and create a LTK. Authenticator SHALL only allow connections from previously bonded clients without user intervention.
5. Client checks if the `fidoServiceRevisionBitfiel` characteristic is present. If so, the client selects a supported version by writing a value with a single bit set.
6. Client reads the `fidoControlPointLength` characteristic.
7. Client registers for notifications on the `fidoStatus` characteristic.
8. Optionally, the client writes a CANCEL command to the `fidoControlPoint` characteristic to cancel the pending request.
9. Authenticator evaluates the request and responds by sending notifications over `fidoStatus` characteristic.
10. The protocol completes when either:
    - The client unregisters for notifications on the `fidoStatus` characteristic, or;
    - The connection times out and is closed by the authenticator. [24]

Commands

PING	(0X81)
KEEPALIVE	(0X82)
MSG	(0X83)
CANCEL	(0Xbe)
ERROR	(0Xbf)

[24]

This protocol commands follow a framing where the messages are written as a single frame (both request and responses). Also, the commands are related with the basic Bluetooth operations that the protocol needs in order to work.

3.1.4 Final considerations

Briefly, the FIDO2 specification lets the final user an easy convenient and simpler way to authenticate themselves to a service using devices that the already own (e.g., cellphone) or by other devices called Authenticators without the usage of passwords. These devices, based on the PKI, allow to enable second factor authentication (with CTAP1/U2F) and first, second or multi-factor authentication (with CTAP2), increasing the security of their personal information and accounts.

Focusing on the complete flow of FIDO2, when the WebAuthn and CTAP protocols join, a complete passwordless environment is enabled. The following image describes the general process:

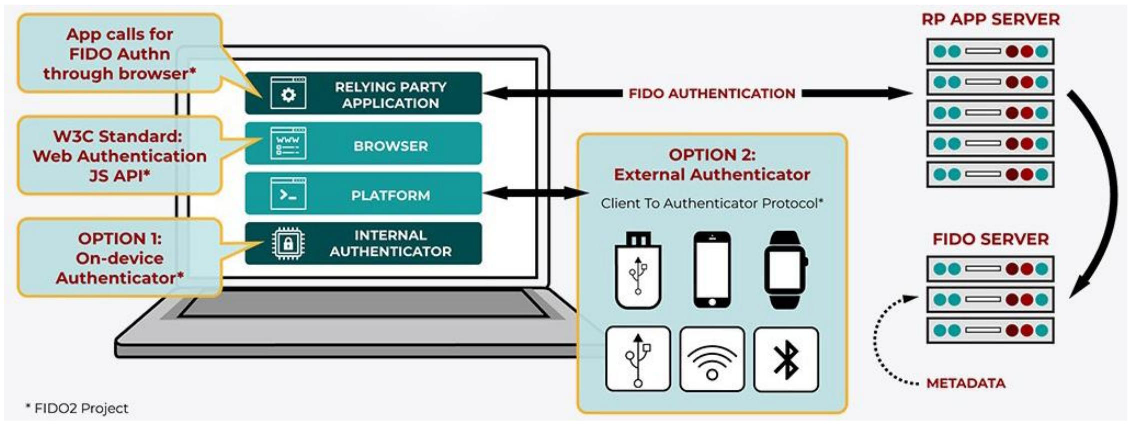

App calls for FIDO Authn through browser*
W3C Standard: Web Authentication JS API*
OPTION 1: On-device Authenticator*
RELYING PARTY APPLICATION
BROWSER
PLATFORM
INTERNAL AUTHENTICATOR
FIDO AUTHENTICATION
OPTION 2: External Authenticator
Client To Authenticator Protocol*
RP APP SERVER
FIDO SERVER
METADATA
* FIDO2 Project

Fig. 12. WebAuthn + CTAP Flow [25]

3.2 Design

As the main objective is to define a passwordless system in an educational environment, a university was used as a model. This entity is chosen because it is more likely to have IT solutions for the wide services it has to offer or use (student’s platform, library, cloud storage, financial systems, etc.), the budget is greater than in schools and the constant changes they manage to improve their services.

Taking this into account, a high-level service map of the possible actual state of a university was modeled, from the authentication point of view. The model does not contain specific information on the services as each organization could implement them different or have additional services that do not interfere with the authentication process.

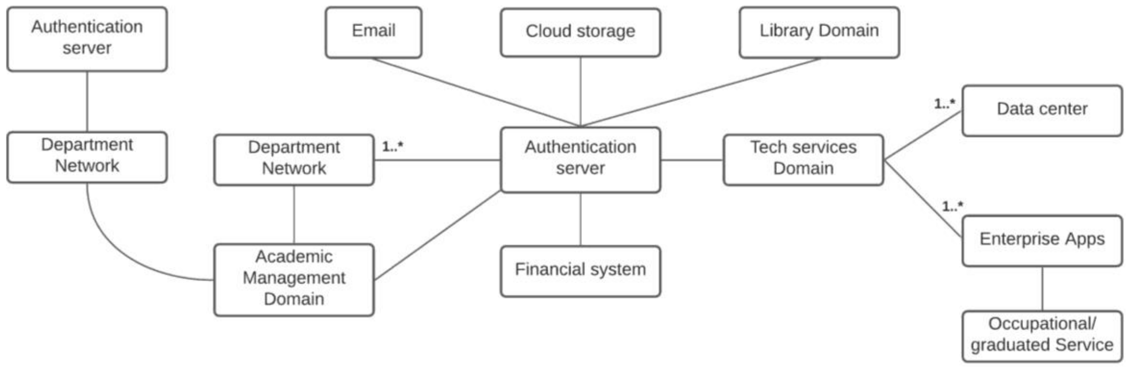


Fig. 13. University service map

Here, every service needs to be in contact with the authentication server which is most likely to provide SSO between those services, as everything needs to be unified. Also, the case when, if needed, a department could have their own subnetwork and implement their own authentication service was considered, considering that it must be linked somehow with the general academic management domain (student’s platform, academic information and records of students are included here).

Now, as the authentication process is centralized, the changes just need to be applied there as when the services require to login, the user is redirected to the authentication server which manages all this process. The initial model does not manage any kind of passwordless authentication; thus, the authentication server consists in a database that stores the encrypted passwords and the user’s information. The following presents a high-level architecture diagram of one service and the authentication process:

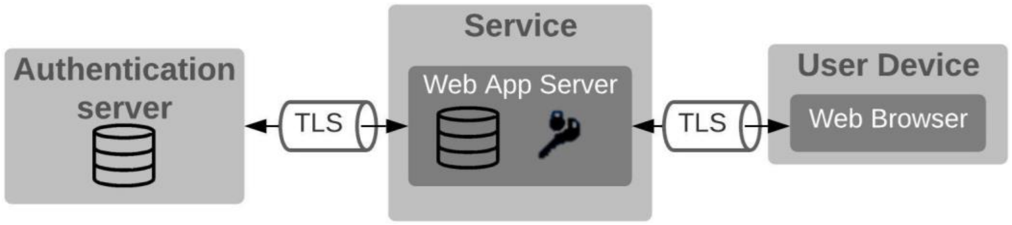


Fig. 14. Legacy High-level architecture

In this context the service and authentication server only present the basic requirements, yet these servers could be as complex as needed. Also, the authentication server could be implemented by the institution or by a third party. Besides, following the web standards, the server provides a session token once the user has logged into the service that will be stored inside the browser while the session is active. This token contains the basic information of the user and its role which each service will have to manage accordingly, for instance, a token could be created with an email and the "student" role.

Now, in order to change to a passwordless environment, the high-level architecture must change to the standards proposed by FIDO2. However, some requirements from the institution must be preserved, for example, the SSO login for their services and the token with the roles. For this, the actual Authentication server must be preserved and modified to adapt the new protocol. Additionally, the user device needs to have access to an Authenticator, it can be internal or external.

Just to follow the specification, the service will be referred as Relying party. The new High-level architecture would be:

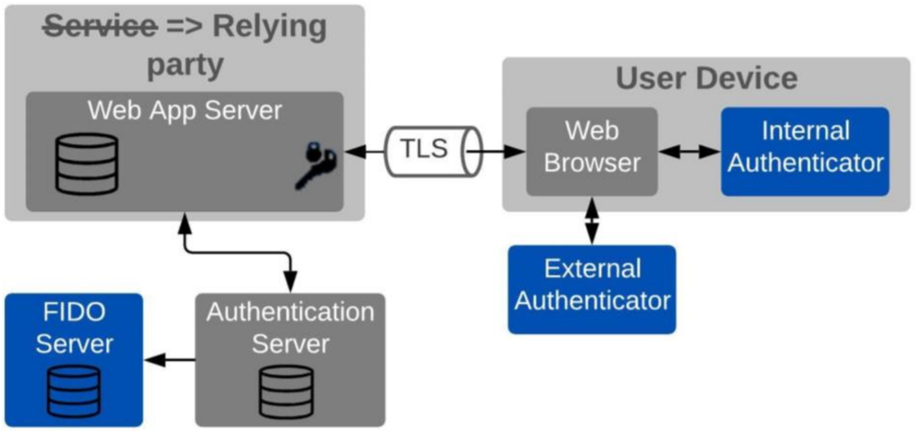


Fig. 15. New high-level architecture

Now, after the architecture is defined, the general authentication protocol must be defined. The protocol has the use cases and the actors.

3.2.1 Services

The following is a description of some services:

- Email

The email service is essential for a university. It allows the communication between all the members with a private email that should be used only for academic/job purposes. The email service is usually given by a third party as it comes with a lot of functionalities as calendar management,

- Academic management domain

This service manages the information of the students. It is one of the main services of a university and needs security measures as this information must be reliable and unmodifiable. Also, it needs roles management for students, teachers and administrative personal.

3.2.2 Actors

- User

The user is the interested in the authentication service, it tries to access the services from a web browser. The user **must** have an Authenticator for the authentication of registration process. The users have defined roles as needed for the university. In this case, the roles of the students and teachers will be defined.

  - Students

  The students can check their grades and graduation requisites on the academic management domain.

  - Teachers

  The teachers can upload grades, manage their courses and check the historical information of their courses.

- User device

The user device must have a web browser that supports the FIDO specifications. The list of the supported browsers and platforms is specified by the FIDO Alliance up to 6/29/2020. This list includes the supported protocols.

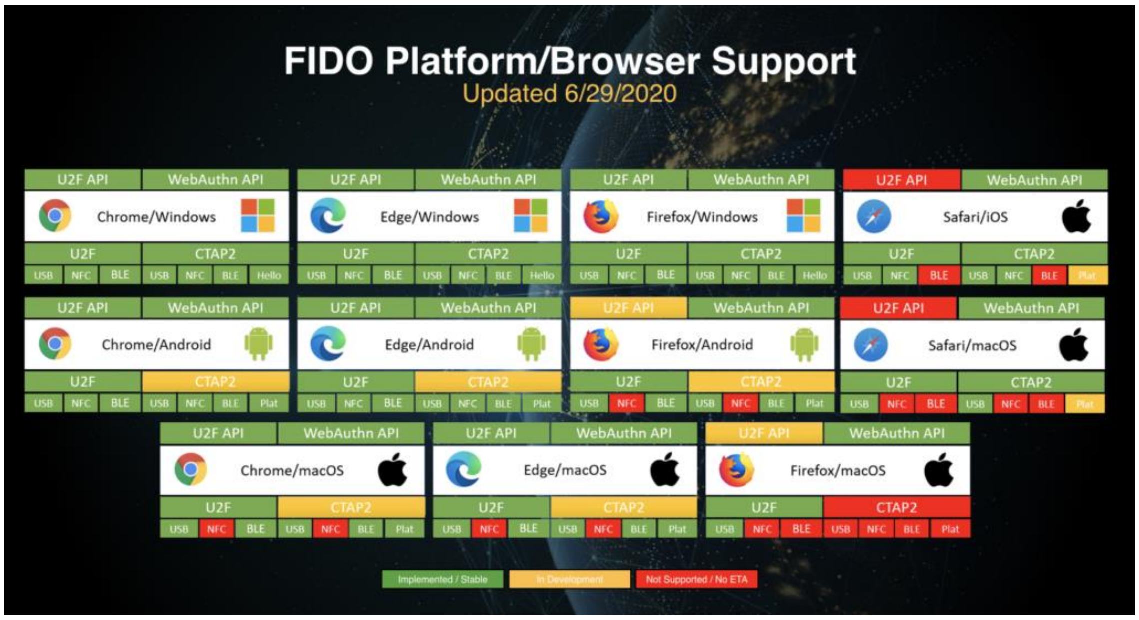


Fig. 16. FIDO Platform/Browser Support [26]

- Relying party

Each one of the services are a RP. When the services require to identify the user, it will redirect them to the Authentication server, which will manage all the user's profile related tasks.

- Authentication server

The Authentication server manages the Authentication and user's profile. It is divided in two servers, the legacy authentication server and the FIDO server. The first maintain the previous information regarding the user and their roles, while the FIDO server verifies the authenticity of the Authenticators used on the process.

- Authenticator

The authenticator is the device that the user uses to authenticate itself. The authenticator must be certified by FIDO and registered for each user.

3.2.3 Use cases

The usage scenarios of a passwordless environment supported by FIDO are the registration, authentication, deregistration and Secure transaction Confirmation. However, the last will not be needed as a university does not need to manage transactions as other institutions like financial or medical.

- Registration

Figure 17 shows the registration procedure. When a **new user** enters the university, it must be registered by the system administrator first (1). In this process, the administrator must enter the information of the new user and assign its roles in the administration console of the authentication server. After this, a unique username and a temporal password is assigned and sent to the user (2).

When the user attempts to login for the first time (3), the system will require him to register a new Authenticator and to change the temporal password (4). For this the user must present the Authenticator provided by the university or its own cellphone/laptop if it is compatible with FIDO2 specifications and has a pre-registered Authenticator such as fingerprints. If the user chooses to use its personal cellphone/laptop, the device will ask him/her to complete the pre-registered authentication process (5.1).

In contrast, if the user is using the token provided by the university, a window will prompt (depending on the OS) where the user will be asked to insert the token via USB (5.2.1). Once the OS recognizes the token, the user will be asked to register the token in the device, this implies setting a pin to the token (5.2.2). Later, the user is asked to press the button on the token (5.2.3) in order to confirm and end the registration process.

If the process is successful, the user can continue using the university services. If not, it must start the registration process again.

Now, if **an existing** user wants to change or register a new Authenticator, the user must contact via telephone or email with the IT administrative department. There, the user can ask either deleting the previously registered device or enabling an additional Authenticator.

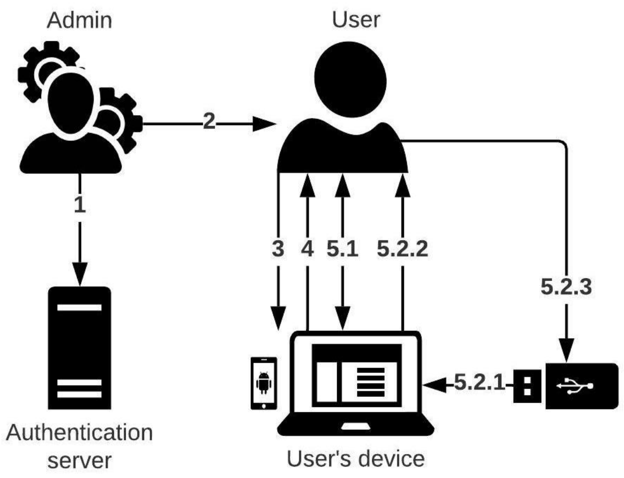


Fig. 17. Registration proposed flow.

- Authentication

Figure 18 shows the authentication procedure. When a user accesses one of the services and tries to log in, the system will require him/her to provide its unique username (1). Then, the systems check if the user has already registered an Authenticator, if not, the device registration process will trigger. Else, the system will start the authentication process (2).

If the user logs in from their personal device compatible with FIDO, the device will ask him/her to complete the pre-registered authentication process (3.1). Otherwise, if the user is using the token provided by the university, a window will prompt asking the user to connect the device if it is not already plug in (3.2.1). When the system recognizes the token, the user must provide the pin (3.2.2) and press the button on the device (3.2.3), if the token matches the registered on the system, the login ends successfully (4). If the token did not match, the system will require the user to present the correct token.

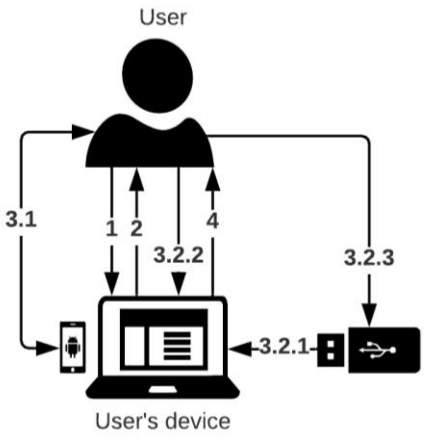


Fig. 18. Authentication proposed flow.

- Deregistration

The deregistration process is done by the system administrator. If a user requires to deregister a device, first their identity must be confirmed. Then, the administrator logs into the authentication server, searches the user and deletes the requested authenticator from the system.

If the user has no other authenticators registered, the administrator must inform the user that the registration process must be done again, and if needed, a temporal password is assigned.

If the user will be deleted from the system, the university politics regarding the personal information must be check before deleting the information, and if needed a backup must be made. After this, the administrator logs into the authentication server, searches the user and deletes its information.

4 IMPLEMENTATION

This section presents the prototype developed to better understand advantages and scope of the design proposed in the previous chapter. However, the whole system was not implemented as there is a time limit to develop such prototype and a lot of services and requirements; a complete implementation would need a team to develop, deploy and maintain. In the design section, a general description was provided and in order to correctly implement the infrastructure some points need to be detailed. For instance, the authentication server must be deployed, this includes defining a federated protocol that allows SSO and defining roles for each of the users. Moreover, the security requirements and use cases need to be developed considering the mentioned actors.

Figure 20 presents the main components of the prototype and their deployment. Users would start their authentication when they access the front-end service from their web browser. Here, according to the indications of the back-end server, the front-end will redirect the user to the Authentication Server. There, the user must authenticate itself to the IdP hosted in the server using the defined authentication method. Once the user has authenticated a cookie is saved in their web browser, and the state of the user is saved in the Authentication server. Then the Authentication server sends to the client the redirection request to take them back to the front-end service. This redirection request contains the authentication response, and in the case the authentication was successful, the information about the user that the two servers previously defined.

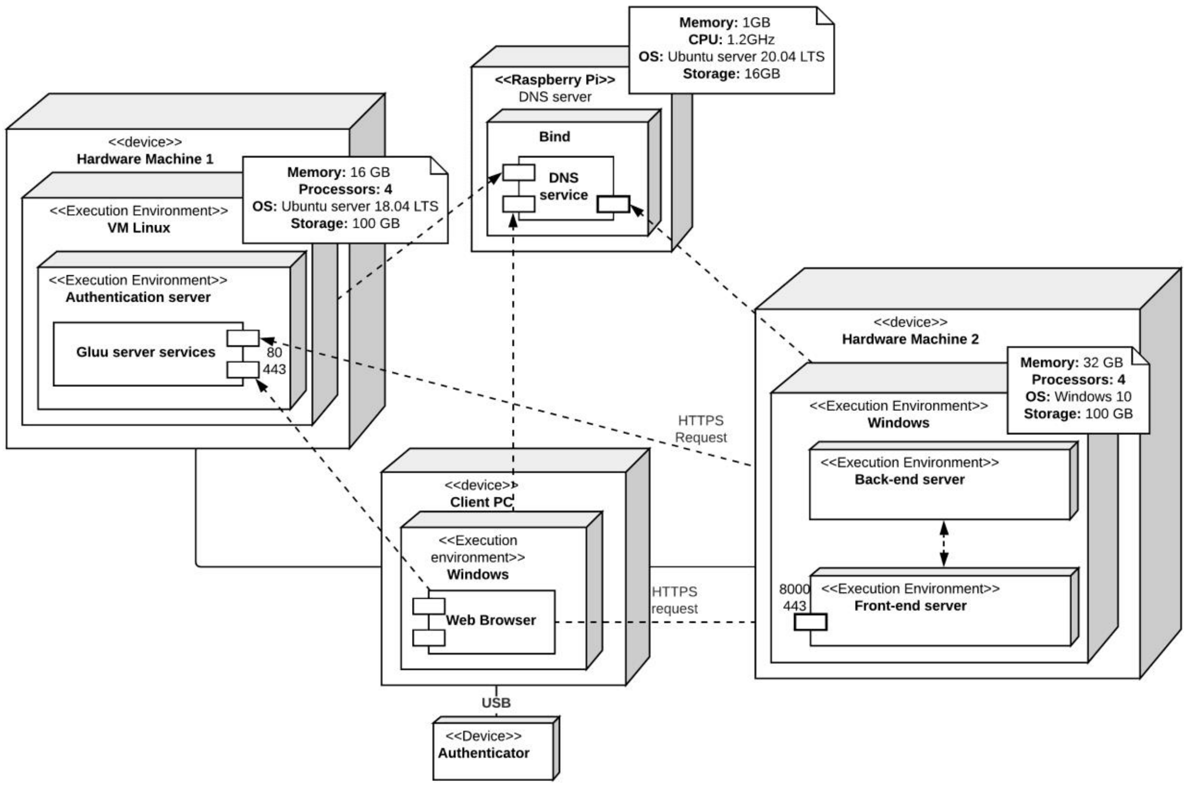


Fig. 19. Solution deployment diagram

First, the SSO schema was designed to be used with passwords. However, related investigations have found that the protocol can still be used with passwordless environments[4] joining the protocols.

[4] Refer to section 6.4 Research on Integrated Authentication Using Passwordless Authentication Method

This schema has two possible options the Security Assertion Markup Language (SAML) and OpenID Connect (OIDC).

- OIDC

  "OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 protocol" [27] that was developed in order to provide the SSO functionality using a centralized Authorization server. This specification is available from multiple devices and platforms (i.e., mobile applications, web browsers, etc.) and provides all the functionalities of Oauth 1.0 and Oauth 2.0. Also, it is used by major platforms like google, Facebook, etc.

  Specifically, as stated on the Oauth explanation video by Okta Developers in the Oauth official website [28], it is a protocol widely used on web development that allows the user to authenticate using a third-party Authenticator (e.g., google) without sending the password to the service. It is mostly used for *access delegation* which allows to share personal information from a different account into a service.

- SAML

  It is an open standard that allows to exchange authorization credentials between *providers*. It is mostly used for SSO schemas and it is based on trust relationships between the *service provider* and the *identity provider*. Also, "SAML transactions use the Extensible Markup Language (XML) for standardized communications between the identity provider and service providers." [29]

Even though OAuth is relatively newer than SAML, they have focus on different points. OAuth is used to simplify the users SSO process and provides *access delegation* across multiple platforms. While, SAML has improved and developed their standard focusing on web services and improved security as the Authentication process does not rely on third-party providers like google or Facebook because it can be implemented inside the enterprise which will manage all the user's processes (it does not manage *access delegation*). For this reason, SAML standard will be used and explained detailed on the implementation.

As mentioned before SAML is a federated authentication protocol that allows SSO between services, currently the managed version is SAML 2.0. It is based on the idea of centralizing the Authentication service of an enterprise with the objective of having all the user's credentials and permissions in one place. Therefore, reducing the possible attack points for a potential cracker to attack, reducing the number of accounts of a user and with that the number of passwords he/she must remember. Additionally, the user experience is improved as the user does not have to login multiple times in the services that are covered by the SSO login.

The following information is provided by the official SAML documentation [30]:

Primarily, the standard defines two roles the *identity provider (IdP)* and the *service provider (SP)*, which have what is called a *trust relationship* where the service trust that the identity provider has the control over the credentials and permissions. Between those two entities, the communication is made via SAML requests and responses (called *assertions*). Finally, the log in can be initiated in two ways, one, the user enters from the IdP and request a login, there it search the wanted service, or the user accesses a service and then he/she triggers the log in process.

Regarding the *trust relationship*, it is achieved using the SAML shared metadata. This metadata is the key to the connection between the IdP and the SP as it defines the identity, the communication method, the protocol endpoints and the cryptographic algorithms that will be used. The identity is checked using the web public certificates and the unique id of the IdP/SP, also, the messages can be signed in order to prove authenticity of the SAML assertion. After both parts have defined their metadata, they share the metadata. This can be achieved by creating a public endpoint with the metadata or by sharing the file directly in the servers.

The SAML 2.0 standard supports the following use cases: IdP initiated SSO, SP initiated SSO in and single logout (SLO). This use cases make use of the protocols defined by the standard, the first two use the “Authentication Request Protocol” and the single sign out uses the “Single Logout Protocol”. However, there are other four protocols: “Assertion Query and Request Protocol”, “Artifact Resolution Protocol”, “Name Identifier Management Protocol” and “Name Identifier Mapping Protocol”.

Finally, the standard defines the following bindings: SAML SOAP, Reverse SOAP, HTTP Redirect, HTTP POST, HTTP Artifact and SAML URI Binding. In the case of SSO, the bindings to be used are HTTP Redirect and HTTP POST [31]. Bindings are used to successfully indicate the specific deployment IP address and the port of a determined service or process.

Now, in order to fulfill all these requirements using open-source applications available on the internet, some prototypes were proposed. The first one used the *onelogin* IdP and SPs, with this a SSO environment was successfully deployed using the IdP management console. Nevertheless, the supported options and modifications as a free user to the IdP were limited to the previously configured services, thus, the necessary modifications to add the passwordless authentication using FIDO 2.0 standard over the federated authentication environment were not possible. On the other hand, the open source *onelogin* “java-saml” project was useful as it offered SAML support for the java applications using the Apache Maven build automation tool. This project is fully modifiable but was not used on the final prototype.

The next prototype uses the open source *Gluu* server for the IdP and the services were developed using NodeJS for the back end and React for the front end. All the instructions and requirements to successfully deploy this environment are specified here:

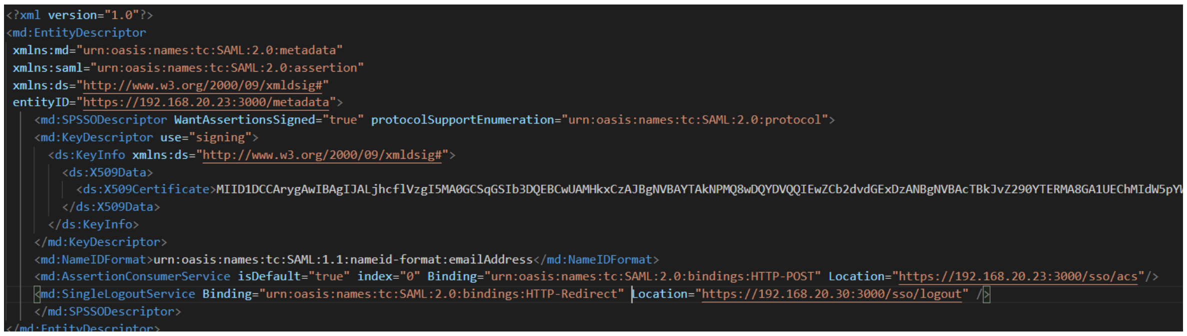
4.1 Service provider

As stated before, the service provider was divided into two servers for backend and frontend services. These servers can be deployed on any platform that supports NodeJS and React, in this case they were deployed on Windows 10 Pro using Visual Studio Code as IDE.

4.1.1 Backend

The backend manages all the communication with the IdP and the session tokens according to the responses of success or fail from the IdP authentication process. These tokens are created using the *jsonwebtoken* Node, which generates them using HMAC with SHA-256. For this, the server needs a private key (that is saved inside the environment variables to prevent possible privilege escalation) a payload and an expiration time. The payload is composed by the user “nameID” and the user’s role which are provided by the IdP in the SAML assertion, while the expiration time is of 1 hour for the email service and 30 minutes for the Academic management service. Additionally, to assure confidentiality over the network, the server must run over HTTPS. For this, the server needs to have its own private key and certificate.

Finally, in order to enable SAML the service metadata must be created, and the server **must** have access to the IdP metadata (either by sharing the file or using a public endpoint). Now, the SP metadata contains the service “entityID”, the X509 certificate information, the nameID format and the defined bindings. The next is an example of the metadata of one of the services:



```xml
<?xml version="1.0"?>
<md:EntityDescriptor
 xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
 xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 entityID="https://192.168.20.23:3000/metadata">
    <md:SPSSODescriptor WantAssertionsSigned="true" protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <md:KeyDescriptor use="signing">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509Certificate>MIID1DCCArygAwIBAgIJALjhcflVzgI5MA0GCSqGSIb3DQEBCwUAMHkxCzAJBgNVBAYTAkNPMQ8wDQYDVQQIEwZCb2dvdGExDzANBgNVBAcTBkJvZ290YTERMA8GA1UEChMIdW5pYW
        </ds:X509Data>
      </ds:KeyInfo>
    </md:KeyDescriptor>
    <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</md:NameIDFormat>
    <md:AssertionConsumerService isDefault="true" index="0" Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location="https://192.168.20.23:3000/sso/acs"/>
    <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" Location="https://192.168.20.30:3000/sso/logout" />
    </md:SPSSODescriptor>
</md:EntityDescriptor>
```

Fig. 20. Service provider metadata example

The service entityID is the endpoint where the metadata of the SP is deployed and available for the IdP to check. Next, the X509Certificate can be found on the certificate generated for the HTTPS connection and include it in the metadata. On the other hand, the nameID format is the user's unique identifier in the system and must be stablished on the configuration of the IdP. These formats are predefined and can be checked on the metadata specifications of SAML, in this case, the identifier is the email address of the user, hence, the value of the format is:

```
urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
```

Continuing, the bindings defined are two. The bindings indicate the IdP where the SAML response Assertions **must** be made and the type of action. In this case the IdP supports the authentication and logout service, thus two directions were specified.

- Authentication

  This case covers the IdP initiated SSO and SP initiated SSO. This direction is called "Assertion Consumer Service" and it is the endpoint where the IdP sends the information after the user successfully logs in.

- Logout

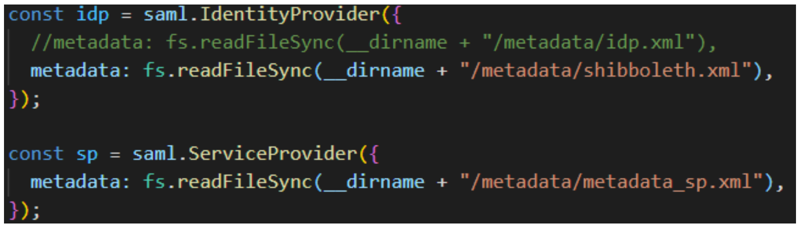
  This case covers the single logout (SLO) use case. This direction is where the IdP will redirect the user after it successfully logs out of service and the IdP.

Once the metadata file is completed, the endpoints must be configured. Two of the endpoints were mentioned but there are more as the frontend needs some extra functions. The following list describes the minimum routes and their responsibilities:

- **Metadata:** this endpoint exposes the SP SAML metadata, that is needed by the IdP, unless every time a modification is made the file is shared with the IdP.
- **Assertion consumer service:** The SAML response is published by the IdP on this route at the end of the authentication process. Here, the server must parse and verify the response, extract the information sent by the IdP and with this, generate the authorization token. Then the user must be redirected back to the front-end service with the generated token, there the user can continue to use the service as usual.
- **SLO:** This route starts the SAML logout redirect. Here the back end must redirect the user to the defined logout direction in the IdP.

Now, the final configuration is to manage the SAML assertions. For this, the node *samlify* was used as it manages the login assertions and the SAML metadata. Therefore, as it does not deal with the logout protocol, it was implemented on the server. The *samlify* library needs the metadata information to manage the assertions, this metadata is loaded using the files created for the SP and

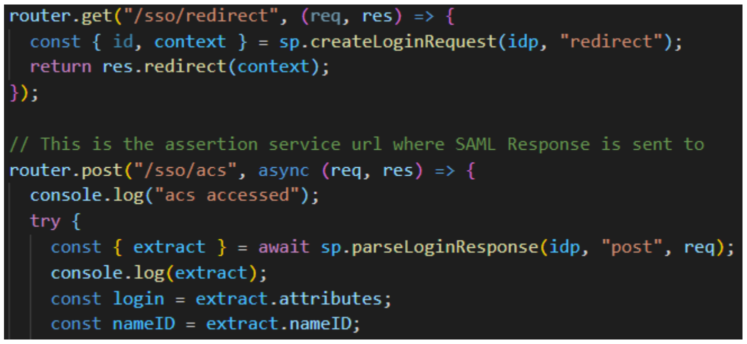
IdP. However, the IdP metadata file could not be read synchronously therefore the file must be shared when a modification occurs, the following image shows the creation of the entities:



```
const idp = saml.IdentityProvider({
  //metadata: fs.readFileSync(__dirname + "/metadata/idp.xml"),
  metadata: fs.readFileSync(__dirname + "/metadata/shibboleth.xml"),
});

const sp = saml.ServiceProvider({
  metadata: fs.readFileSync(__dirname + "/metadata/metadata_sp.xml"),
});
```

Fig. 21. *samlify* entities

Using these entities, the library is capable of create the SAML Request that are send to the service provider. Additionally, it manages the SAML Responses, validates them and allows the extraction of the fields needed to continue the process. The following image shows the creation of the login request and the reception of the login response sent by the IdP.



```
router.get("/sso/redirect", (req, res) => {
  const { id, context } = sp.createLoginRequest(idp, "redirect");
  return res.redirect(context);
});

// This is the assertion service url where SAML Response is sent to
router.post("/sso/acs", async (req, res) => {
  console.log("acs accessed");
  try {
    const { extract } = await sp.parseLoginResponse(idp, "post", req);
    console.log(extract);
    const login = extract.attributes;
    const nameID = extract.nameID;
```

Fig. 22. *samlify* request and response

Once the information sent by the IdP is parsed and extracted, the backend creates the token as specified before and sends it to the frontend where the user continues using the service.

4.1.2 Frontend

The front end manages the interaction with the user and consumes the tokens generated by the backend service to manage all the user information. However, the frontend does not know the content of the token, if it needs to access the user information it sends a request to the backend using the token as Authorization method. In this case, the frontend visualization uses the bootstrap Framework, and the routing uses the *react-router* n-ode. Additionally, each time a client logs in/out, the application must save/delete the token generated by the backend service into the browser to maintain the user's session. This is achieved using the local storage of the browser.

The visualization pages are simple with the necessary options to continue the authentication process between the applications. This is the example of the login page:

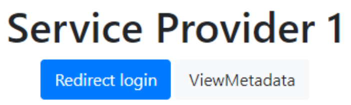


Fig. 23. Service provider home view

As the backend manages all the assertions, the frontend only has to redirect and inform the user about the process. However, as the authentication process is provided by the IdP, those views are modified there.

Finally, some of the “roles” process is managed by the frontend. Every time a user logs in, the service asks the backend for the roles and with this information it shows or hides the views from the user. Even though this configuration is made, when an action is completed, the backend must authenticate the user’s role and then complete the action.

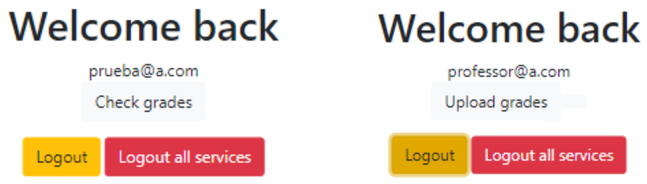


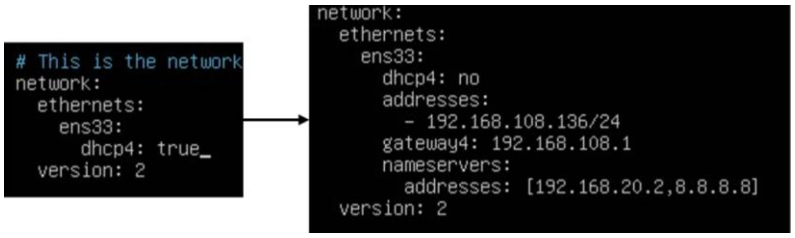
Fig. 24. Roles views

4.2 Identity provider

The identity provider was deployed using the *Gluu* authentication server. This server was deployed on an Ubuntu Server 18.04.5 virtual machine running on VMware with the following specifications:

Specification	Used	Recommended
CPU unit	4	2
RAM	16 GB	8 GB
Disk Space	120 GB	40GB
Processor Type	64 Bit	64 Bit

Additionally, the server needs to have configured a static IP and a domain. The IP is configured changing the *netplan* file settings, where, the IP address, the gateway, the nameservers and the dhcp4 must be changed. This file is located on the `/etc/netplan` directory. The following image shows the state of the file before and after the static IP is set[5].

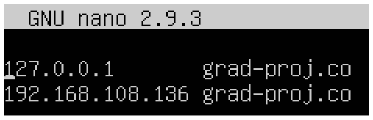


```
# This is the network
network:
  ethernets:
    ens33:
      dhcp4: true_
  version: 2
```

```
network:
  ethernets:
    ens33:
      dhcp4: no
      addresses:
        - 192.168.108.136/24
      gateway4: 192.168.108.1
      nameservers:
        addresses: [192.168.20.2,8.8.8.8]
  version: 2
```

Fig. 25. VM network configuration file

The hosts file must be modified too (it is located on `/etc` directory), adding the name of the domain that will contain the IdP service. The following is an example of the configuration file:



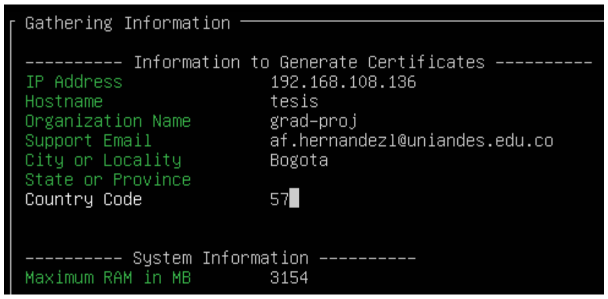
```
  GNU nano 2.9.3

127.0.0.1       grad-proj.co
192.168.108.136 grad-proj.co
```

Fig. 26. VM hosts configuration file.

[5] The IP directions depend on the used network, thus, to configure it the router configuration must be checked first.

Now, to install the server, G*luu* provides a guide that contains information regarding the system specifications and details on the installation [32]. However, as the server contains a lot of services, the setup script configuration will be described.

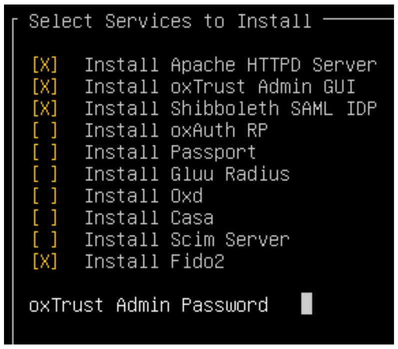
The script checks the available memory and ask for the acknowledge of the Apache-2.0 license. After this, the information to generate the certificates will be asked. This form must be filled:



```
┌ Gathering Information ─────────────────────────────────────

  ---------- Information to Generate Certificates ----------
  IP Address                 192.168.108.136
  Hostname                   tesis
  Organization Name          grad-proj
  Support Email              af.hernandez1@uniandes.edu.co
  City or Locality           Bogota
  State or Province
  Country Code               57


  ---------- System Information ----------
  Maximum RAM in MB          3154
```

Fig. 27. Certificates information

After this, the script will ask the services that will be installed. The following **must** be selected:

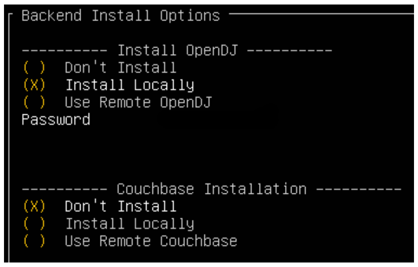


```
┌ Select Services to Install ──────

  [X]  Install Apache HTTPD Server
  [X]  Install oxTrust Admin GUI
  [X]  Install Shibboleth SAML IDP
  [ ]  Install oxAuth RP
  [ ]  Install Passport
  [ ]  Install Gluu Radius
  [ ]  Install Oxd
  [ ]  Install Casa
  [ ]  Install Scim Server
  [X]  Install Fido2

  oxTrust Admin Password
```

Fig. 28. Gluu service selection

The Apache HTTPD allows to use manage console over HTTP instead of just modifying the files, also it allows web services for the identity management. The oxTrust Admin GUI is the Gluu server administration service, here the administrator can manage all the other services and the users' information. The Shibboleth SAML IDP provides all the services that the open-source project has available, using the oxTrust interface and the customizable scripts. This service manages all the SAML related processes, which include the metadata, assertions, etc. Lastly, the FIDO2 extension manages all the passwordless or multifactor authentication related processes.

And the persistence mechanism must be chosen, for this the default is OpenDJ which works with the LDAP standard. It will be installed locally, and s password must be set.



```
┌ Backend Install Options ──────────────────────

  ---------- Install OpenDJ ----------
  ( )  Don't Install
  (X)  Install Locally
  ( )  Use Remote OpenDJ
  Password



  ---------- Couchbase Installation ----------
  (X)  Don't Install
  ( )  Install Locally
  ( )  Use Remote Couchbase
```

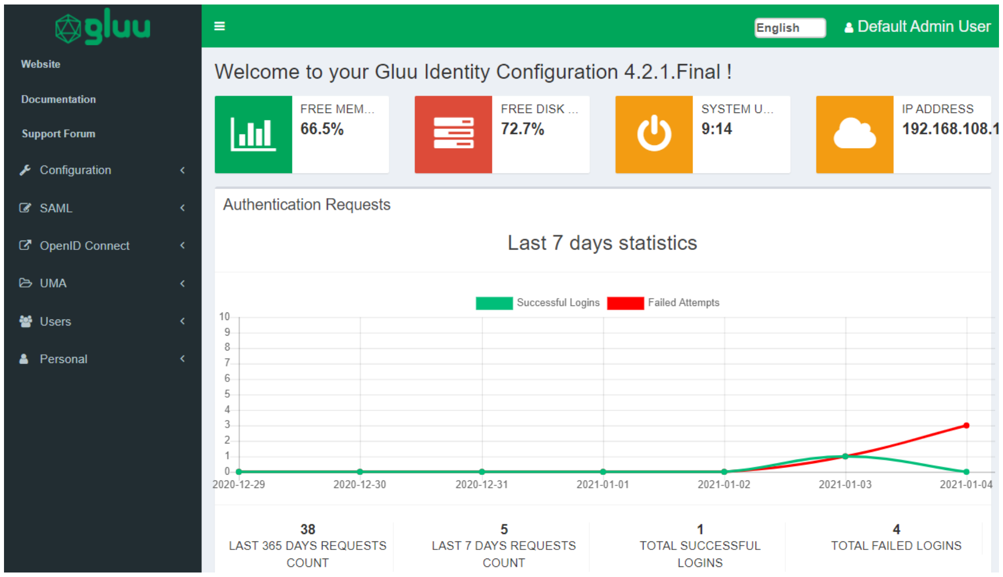
Fig. 29. Gluu persistence decision

Finally, a summary of the installation will prompt, and it will take approximately twenty minutes to complete the installation. After the installation is over, it will take about ten minutes to restart the server and configure everything using the oxTrust administration service.

However, before the administration console is accessed, in order to enter from different devices, the domain must be redirected to the virtual machine IP address. Therefore, a private DNS must be set before hand as it will successfully associate the domain name to the IP address of the server. This DNS can be implemented in multiple ways, and, for this prototype it was deployed using a ubuntu server 18.04.5 with the *bind* service which offers a reliable DNS easily deployed.

Another option to associate the domain name to the IP address can be achieved modifying the hosts file in the wanted device. Unfortunately, not all the devices allow this, or some users could not have permission to do so. In windows, the file can be modified with an administrator account. The file is located on the route "C:\\windows\System32\Drivers\etc\hosts" and can be modified by any text editor run as administrator.

Once the name resolution has been configured, the configuration console is accessed from the domain name defined to the project. There a login will prompt, the user is "admin", and the password is the one that was provided during the setup process. Once the credentials are provided, the administration main page will prompt. This page contains the information about the system, the statistics over the last days or months from the Authentication requests and the services configuration sections.



Welcome to your Gluu Identity Configuration 4.2.1.Final !

FREE MEM... 66.5%
FREE DISK ... 72.7%
SYSTEM U... 9:14
IP ADDRESS 192.168.108.1

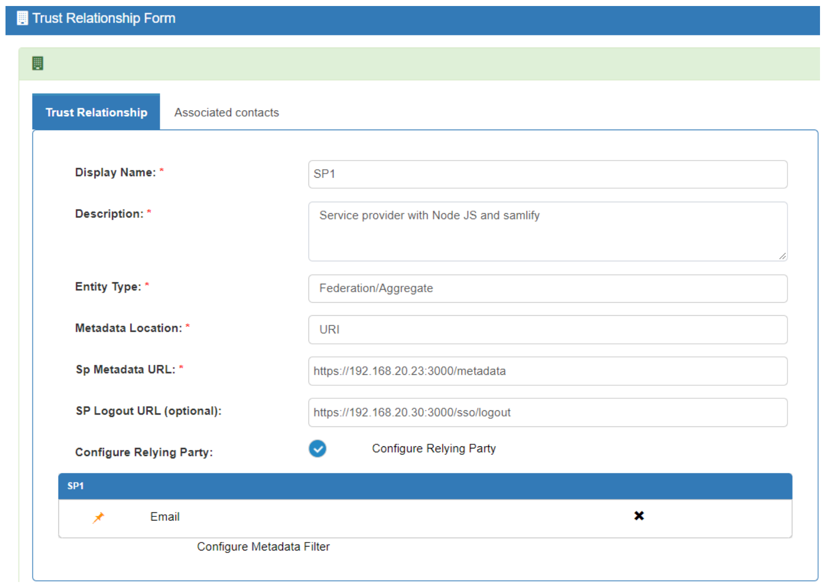
Authentication Requests

Last 7 days statistics

38	5	1	4
LAST 365 DAYS REQUESTS COUNT	LAST 7 DAYS REQUESTS COUNT	TOTAL SUCCESSFUL LOGINS	TOTAL FAILED LOGINS

Fig. 30. Gluu administration main page

Now, before starting the configuration of the passwordless environment, the SAML protocol must be completed. The protocol is managed on the SAML section on the side bar, there, the administrator can configure the trust relationships, add a new one or configure a custom NameID (the unique identifier for the users using the protocol). To configure a trust relationship, the information of the SP must be provided according to the information defined on the SP metadata.

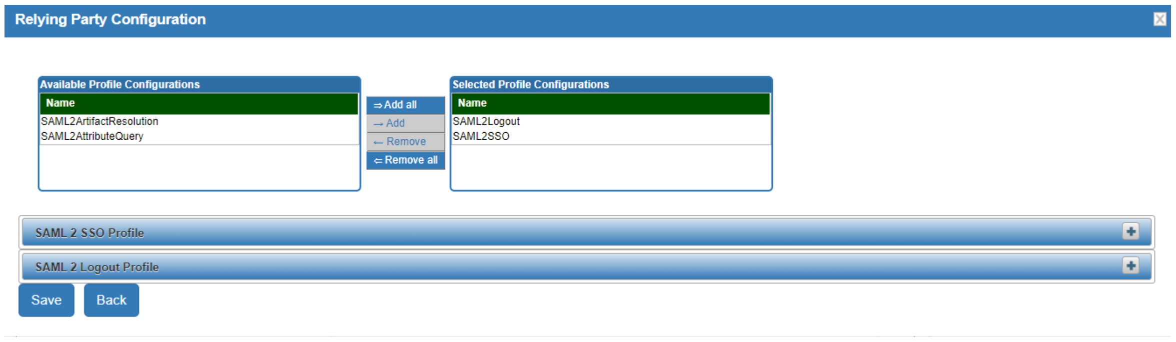


Display Name: *	SP1
Description: *	Service provider with Node JS and samlify
Entity Type: *	Federation/Aggregate
Metadata Location: *	URI
Sp Metadata URL: *	https://192.168.20.23:3000/metadata
SP Logout URL (optional):	https://192.168.20.30:3000/sso/logout
Configure Relying Party:	✓ Configure Relying Party

SP1
Email

Fig. 31. New trust relationship form

The display name and the description are just used to identify the relationship. The entity type can be Federation/Aggregate or a single SP, in this case it is a Federation/Aggregate, the metadata has the URI defined in the backend and the URL where the SLO response is send. Moreover, the protocols to be used must be defined. For this the checkbox must be selected and click in the configure relying party button. The configuration view is the following:



Available Profile Configurations

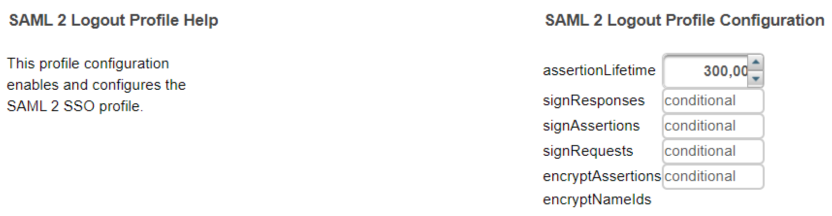
Name
SAML2ArtifactResolution
SAML2AttributeQuery

Selected Profile Configurations

Name
SAML2Logout
SAML2SSO

Fig. 32. Relying Party configuration

Here the two used protocols are selected (Logout and SSO) and the configuration section for each will prompt. In this configuration section the assertion lifetime and the protocol standards are configured. All is set to conditional, which means that the protocol assertions are configured using the exchanged metadata, where it is specified if the assertions must be signed or encrypted or both.


SAML 2 Logout Profile Help

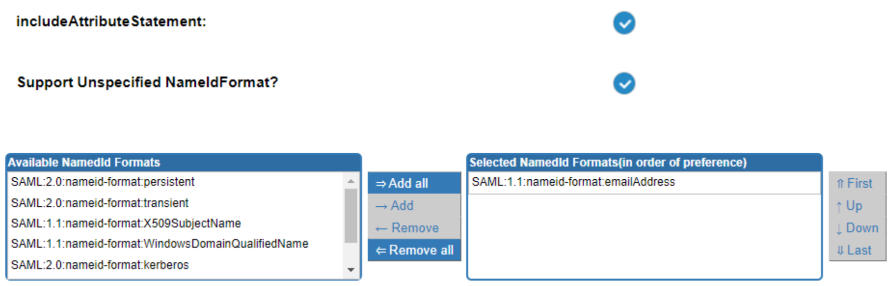
This profile configuration enables and configures the SAML 2 SSO profile.

SAML 2 Logout Profile Configuration

assertionLifetime	300,00
signResponses	conditional
signAssertions	conditional
signRequests	conditional
encryptAssertions	conditional
encryptNameIds	

Fig. 33. Assertion configuration options

The SSO has an additional configuration where the nameID format must be specified, there we select the same as the SP email address.



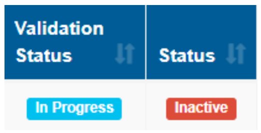
includeAttributeStatement: ✓

Support Unspecified NameIdFormat? ✓

Available NameId Formats		Selected NameId Formats(in order of preference)	
SAML:2.0:nameid-format:persistent	⇒ Add all	SAML:1.1:nameid-format:emailAddress	⇑ First
SAML:2.0:nameid-format:transient	→ Add		↑ Up
SAML:1.1:nameid-format:X509SubjectName	← Remove		↓ Down
SAML:1.1:nameid-format:WindowsDomainQualifiedName	⇐ Remove all		⇓ Last
SAML:2.0:nameid-format:kerberos			

Fig. 34. SSO configuration options

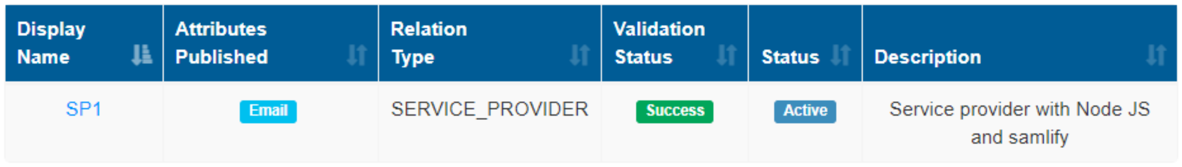
After filling the form, the trust relationship is left on the status “inactive” and the validation status is set “in progress”.



Validation Status	Status
In Progress	Inactive

Fig. 35. SP status in the IdP

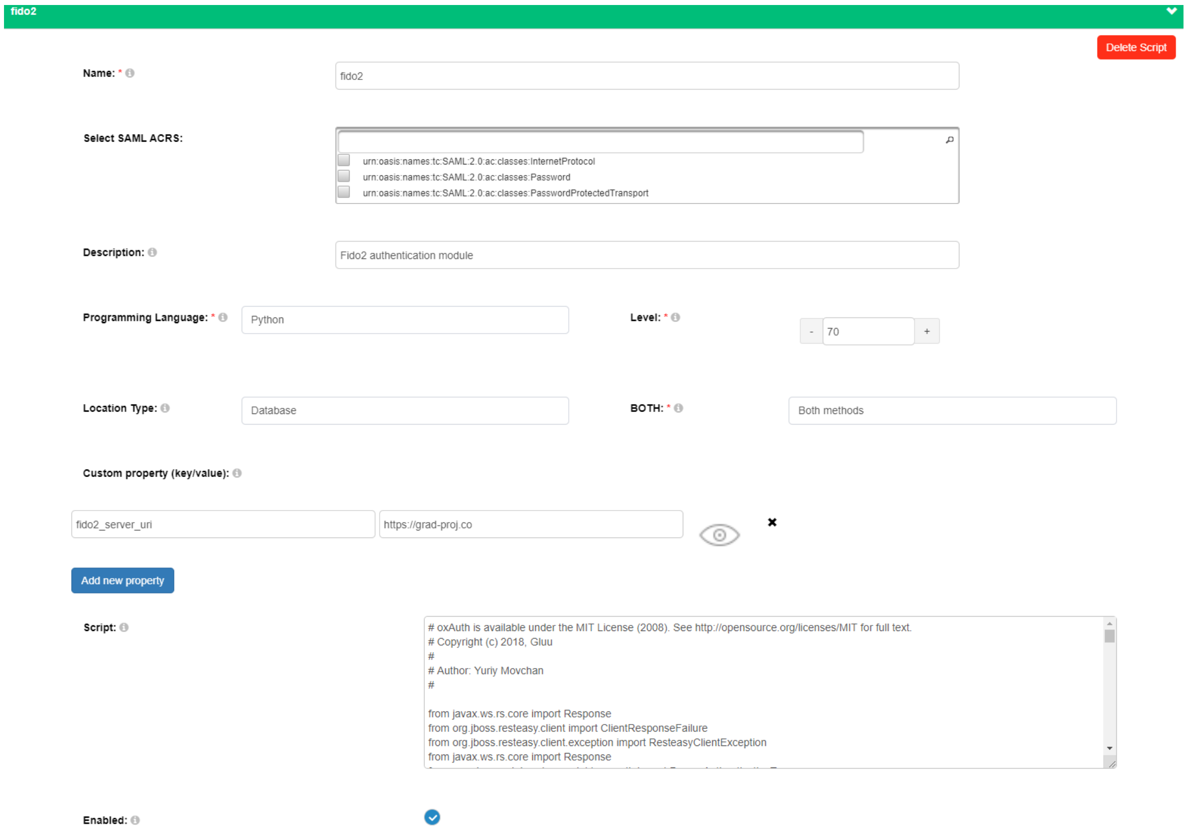
To activate the new trust relationship, the new SP configuration must be accessed and the “activate” button clicked. After this, it will take around five minutes for the service to restart. Once that is setup, the validation is performed when the first user logs in to the service successfully. The status and the validation status should change, if not, check the logs backend logs or the IdP logs to correct the metadata or the IdP configuration.



Display Name	Attributes Published	Relation Type	Validation Status	Status	Description
SP1	Email	SERVICE_PROVIDER	Success	Active	Service provider with Node JS and samlify

Fig. 36. Service after successful configuration

Now that the initial configuration is completed, the FIDO2 standard has to be included in the process. The Gluu server supports all the standards of the FIDO Alliance (U2F, UAF and FIDO2) and each one has a different configuration. These and other authentication standards are managed via “custom scripts” that are developed by the server admin mostly in python. However, the server has examples of this scripts that are ready to use. In order to configure and deploy the FIDO2 passwordless authentication, access the “configuration”→”Person Authentication Scripts” page and search for the fido2 option. There, the following or a similar view should prompt:



Name: * fido2

Select SAML ACRS:
- [ ] urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocol
- [ ] urn:oasis:names:tc:SAML:2.0:ac:classes:Password
- [ ] urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport

Description: Fido2 authentication module

Programming Language: * Python

Level: * 70

Location Type: Database

BOTH: * Both methods

Custom property (key/value): fido2_server_uri https://grad-proj.co

Script:

```
# oxAuth is available under the MIT License (2008). See http://opensource.org/licenses/MIT for full text.
# Copyright (c) 2018, Gluu
#
# Author: Yuriy Movchan
#

from javax.ws.rs.core import Response
from org.jboss.resteasy.client import ClientResponseFailure
from org.jboss.resteasy.client.exception import ResteasyClientException
from javax.ws.rs.core import Response
```

Enabled: ✓

Fig. 37. Fido2 IdP script view

The “enable” checkbox must be selected, in “BOTH” “Both methods” must be selected in order to provide access from web applications and from native applications. The example script provided replaces the authentication process of SAML, there the admin must indicate the authentication process. By default, the script allows second factor authentication using FIDO2 standard, but multi-factor authentication or passwordless authentication can easily deployed modifying this script and adding the corresponding views.

However, this only activates the feature, it does not change the actual authentication method used by SAML. For this, the FIDO2 endpoints must be enable on the “Configuration” →”JSON configuration” page, “FIDO 2 Configuration” tab. There, search the “disable” checkbox and set it to “False”. After this, the `mdsAccessToken` must be provided and at the end of the page click the “Save configuration” button.

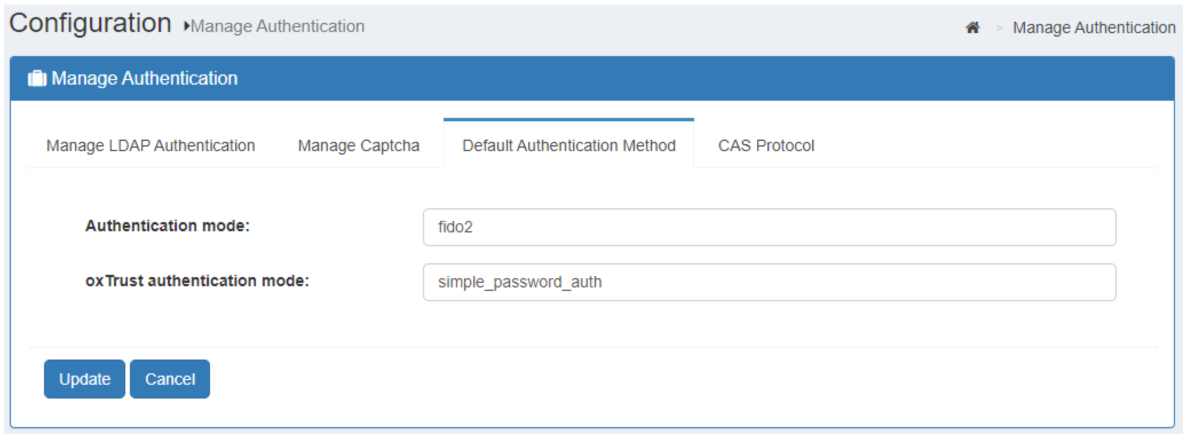
The `mdsAccessToken` is a token provided by the FIDO alliance in order to access the FIDO Server (refer to the New High-level Infrastructure diagram). In this server the FIDO alliance provides the Metadata Service (MDS) which provides all the interested organizations a centralized and trusted server with the information of all the registered FIDO Authenticators. This information is used to protect the users as it has the information regarding new discovered vulnerabilities that the user should be aware of as the device could need a software update or to be changed.

To get a token, the admin must register to the MDS service or renew the expired token if it has one. This process is done directly on the FIDO alliance page on this link: https://mds2.fidoalliance.org/tokens/. After registering, a confirmation email is received. Then, a mail containing the token, expiration date and a copy the usage terms is send to the registered email.

After filling the `mdsAccessToken` field and saving the configuration, the admin must download the last Table of Contents (TOC) file to the server. This file is obtained using the `mdsAccessToken` and it is used to verify the Authenticators each time an assertion is made. To download the file, the admin must be log into the chroot of the Gluu server, this is the commands to download the TOC file, if the admin is already logged into the chroot, skip the first command.

```
/sbin/gluu-serverd login

cd /etc/Gluu/conf/fido2/mds/toc

wget https://mds.fidoalliance.org/?token=<YOUR_TOKEN> - O toc.jwt

cd /etc/Gluu/conf/fido2/mds/cert

wget https://mds.fidoalliance.org/Root.cer

systemctl restart oxauth
```

With this the FIDO authentication process is ready to be integrated with the SAML authentication process and deployed. For this, on the "Configuration"→"Manage Authentication page", "Default Authentication Method" bar, set "Authentication mode" to "fido2".


Configuration ▸Manage Authentication
Manage Authentication
Manage Authentication
Manage LDAP Authentication
Manage Captcha
Default Authentication Method
CAS Protocol
Update
Cancel

Authentication mode: fido2
oxTrust authentication mode: simple_password_auth

Fig. 38. Changing the default authentication method.

5 EVALUATION

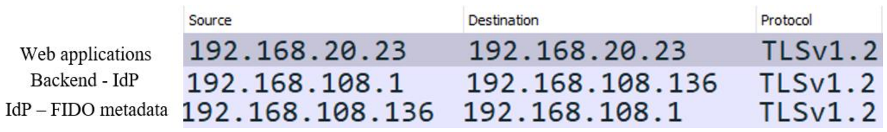
The final prototype offers a passwordless environment, however, it needs to be tested to verify that all the configurations are right and that there are not open ways for an attack. Moreover, some usability tests can determine if the system is better than a password-based system. Finally, a comparation between the two paradigms can point out which one is better on the actual context.

The prototype was tested using an external Authenticator on a desktop with Windows as SO, using Microsoft Edge, Google Chrome and Mozilla Firefox as web browsers. The tests evidenced that there was no difference between browsers. Next, the Authenticator used was a Thetis FIDO certified token which can be plug in via USB.

5.1 Tests results

The first tests were made using the Wireshark tool. This program allows the capture off all the packets that are sent into the network through a specific network interface. Also, it can be configured to decrypt the TLS packets (using Chrome or Firefox) setting a global environment variable called SSLKEYLOGFILE with the route of a text file. In this file, the browser logs all the keys that are result of **all** the key exchange process while the browser is open, and the program uses the file to decrypt the packets.

The traffic captures were taken from the Authentication and registration of a new device protocol (as the registration process of a user is made by the system administrator). The captures show that the four connection points (user device, RP, IdP and FIDO server) are ciphered using the TLS over HTTP protocol:



	Source	Destination	Protocol
Web applications	192.168.20.23	192.168.20.23	TLSv1.2
Backend - IdP	192.168.108.1	192.168.108.136	TLSv1.2
IdP – FIDO metadata	192.168.108.136	192.168.108.1	TLSv1.2

Fig. 39. HTTPS between components

The 20.23 is the IP address of the web application server and the 108.136 is the IP address of the IdP server. As the servers were on different networks, the packets that enter or exit the IdP are always captured in the gateway. Therefore, to identify when the IdP started the communication with the backend or FIDO metadata server the handshake packets were searched to confirm the secure channel creation.

Moreover, the captures allowed to specify the complete protocol, however some actions like the Authenticator communication and operations could not be captured with this tool. Even though, they were included in the protocols of both use cases:

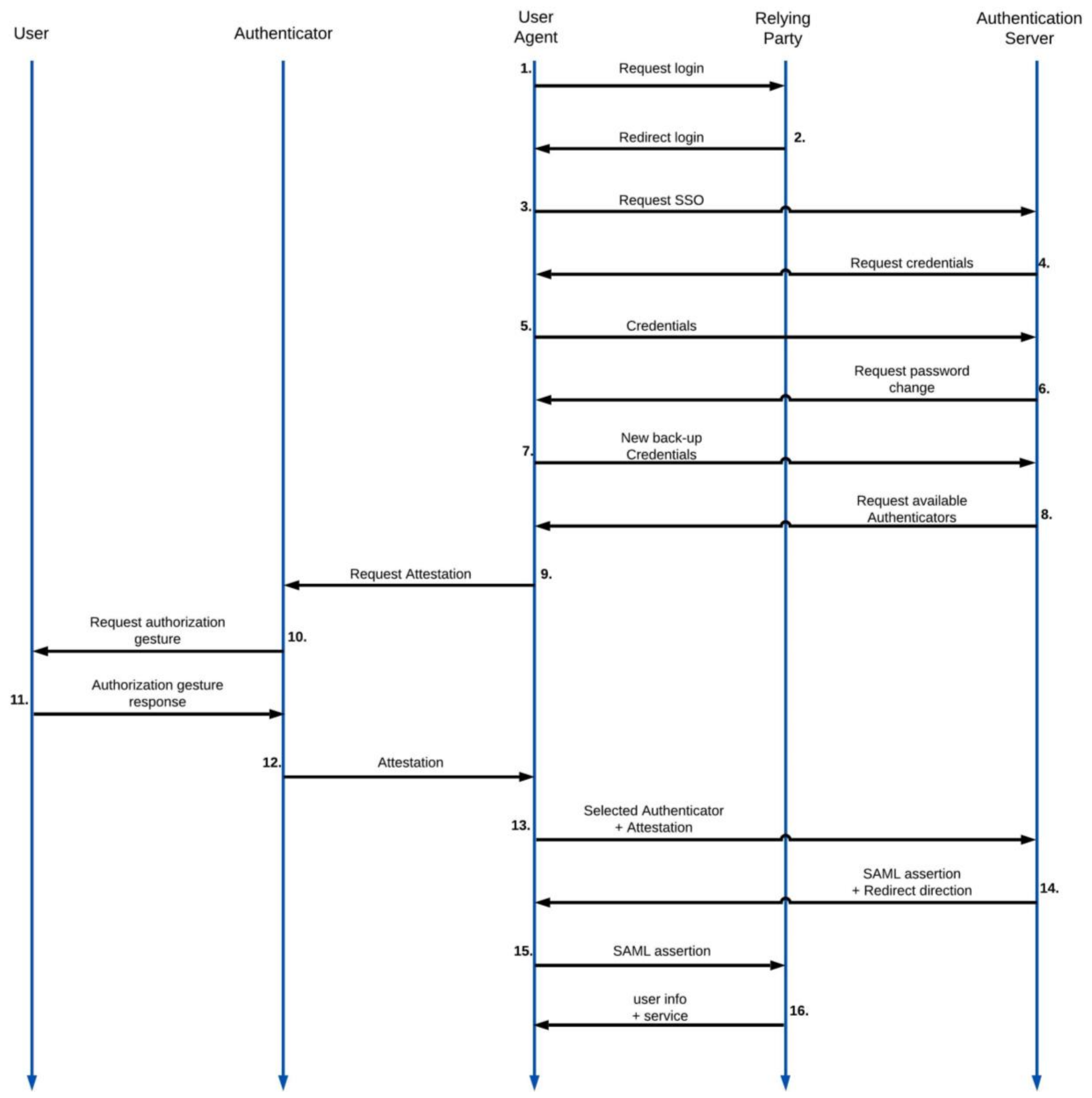

User
Authenticator
User Agent
Relying Party
Authentication Server
1. Request login
2. Redirect login
3. Request SSO
4. Request credentials
5. Credentials
6. Request password change
7. New back-up Credentials
8. Request available Authenticators
9. Request Attestation
10. Request authorization gesture
11. Authorization gesture response
12. Attestation
13. Selected Authenticator + Attestation
14. SAML assertion + Redirect direction
15. SAML assertion
16. user info + service

Fig. 40. Complete Registration protocol

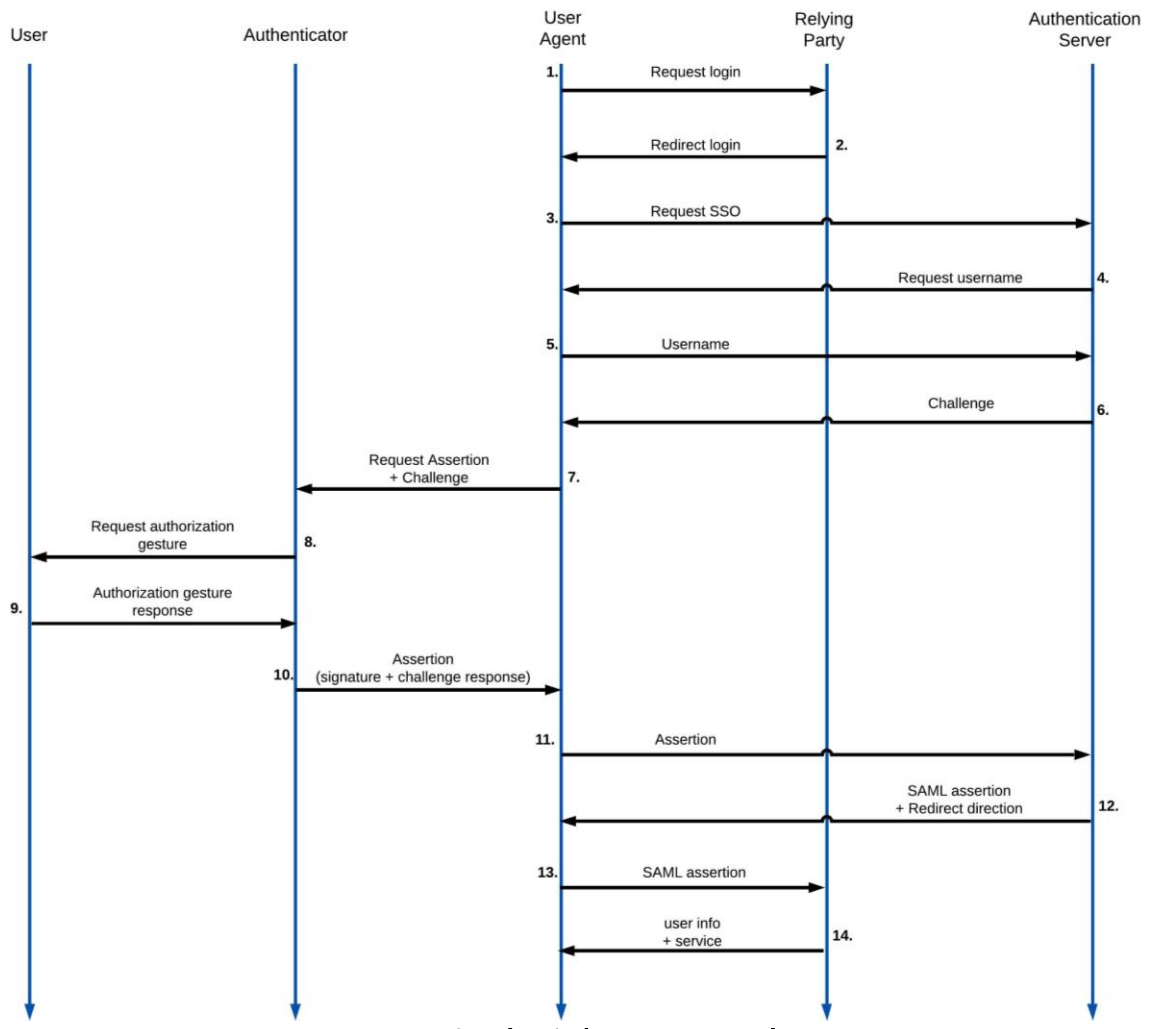


Fig. 41. Complete Authentication protocol

Now, the TLS is not the only encryption system that protects the packets security. The SAML configuration allows to encrypt the user's identifier (NameID) or the complete assertions during the process using the keys included inside the metadata.

However, during the FIDO2 protocols the information transmitted is not encrypted. Thus, the WebAuthn and CTAP protocols rely completely on the fact that the RP stablishes a secure channel during the message exchange. This means that there is a possibility that during the new device registration process the system is vulnerable to a man in the middle attack, here the attacker may intercept and change the package containing the authenticator attestation and access the other services registering its own authenticator.

Even though the FIDO2 standard does not implement the additional encryption, during the process user's personal information is not exposed as this information is sent inside the SAML assertions. The information "exposed" is the assertion containing the metadata and the public key during the registration process, and the response to the cryptographic challenge during the authentication process. Therefore, the prototype provides *confidentiality* as the user's information is not exposed during the authentication processes.

Regarding the Authenticator, the external Authenticator used showed that this device can be used by multiple services as it manages the assertions from different origins with no apparent problem. Interestingly, this device can be used by multiple persons during the authentication process without affecting or exposing the private key of a different service or person. Further investigation about the internal functioning of the device should be made as it is the center of the authentication processes.

Finally, due to problems with the deployment of the IdP over a public IP and limited time, usability test with real users could not be done. The objective of this test was to analyze if the users feel more comfortable and secure using the solution proposal or the legacy environment. Additionally, the users would be asked to think and indicate about possible cases they have been exposed to in the password-based environment to check if the same are still needed in the new paradigm.

Nevertheless, some cases were contemplated as they are not contemplated inside the FIDO2 standard and must be implemented inside the authentication server. First, the standard does not propose a recovery method in case a user wants to change its Authenticator or loses it, which is in fact a common case as the users could change their cellphone, external Authenticator, etc. to update it. This case also leads to the suspension of the accounts, when an Authenticator is compromised, the standard should provide a clear and defined protocol that allows the user and application administrator to block/suspend the related account until the problem has been solved. At last, considering the last two points, highly required services should not be defined with a passwordless environment alone as the user would be completely blocked of the service if an Authenticator is compromised. Therefore, these services need to adopt a back-up password related authentication system or use multifactor authentication with defined acceptance levels that vary the authorization as required.

5.2 Comparation legacy vs proposed system

Removing the passwords on the proposed system means that all password related problems are not a threat anymore as the user does not need to remember anything; the weak passwords, reuse of passwords in different accounts and the ease of access to passwords problems are gone. Furthermore, the attacks from crackers are invalidated as the dictionary attacks, brute force and keyloggers cannot capture any information that could compromise the user's account. Also, the social engineering strategies are mitigated as the user cannot provide useful information to the attackers as they cannot lend biometrics and it is not likely a user would lend their personal devices to a stranger. Now, as the Authenticators are the center of the process, attackers could still try to design malware to access somehow the information stored there. Finally, as the transmission channels could still be compromised, the man in the middle attacks are still a thread, however it is decreased as this attack is only viable when the user is doing the registration process. The following table resumes the cases:

Problems	Legacy	proposed
Passwords	✓	X
Reuse credentials	✓	X
Keyloggers	✓	X
Brute force attacks	✓	X
MIM / traffic interception	✓	✓
Phishing	✓	X
Baiting	✓	✓
Malware	✓	✓
quid quo pro	✓	X

Focusing on the usability of the legacy environment, it is better as the people already know how to proceed on each use case and all the cases are almost standard, however, it has the drawback of the password itself which to be secure is most likely to be forgettable. In contrast, the use cases of the proposed system could vary in each implementation, which could generate confusion among the users as there is not a defined standard. Yet, the main use cases have an advantage as the user does not have to remember anything, just provide their selected Authenticator which is very intuitive and the standard itself suggest including the step by step into the user's interface.

5.3 Achievement of objectives

As stated on the proposal, the main objective of "design, implement and deploy a passwordless system in an educational environment" was met. The developed environment also led to the understanding of the components needed for a passwordless-based system using the FIDO2 standard with a federated protocol such as SAML for the Single Sign-On processes. Besides some advantages as the encryption of all personal data and challenges like the missing use cases on the standard were found/corroborated during the investigation process. Finally, the decrease in the risk of identity fraud objective was met during the comparation of the legacy system with the proposed system.

6 RELATED WORK

The following papers were consulted in order to understand the actual state of FIDO2 in the world, some of the implementations that have been made and research about it. The following sections have the next structure: context, problem, solution and comments.

6.1 Improved Identity Management with verifiable credentials and FIDO [33]

- Context

Description of the functioning of FIDO and W3C VCs and the problems they can solve. Also, a solution extending the protocol "in order to provide both strong authentication and strong authorization". This model was tested in a UK hospital.

- Problem

While FIDO2 provides strong authentication, it does not provide authorization and complete identification.

- Solution

Combined the FIDO UAF and W3C VC architectures and extended the UAF protocol to provide strong authentication and authorization.

- Comments:

They made a good investigation of the functioning of FIDO and company. Which lead them to a secure extended protocol that included a lot of points of view that search to improve the user privacy and security. Most of the mistakes were usability issues that can be easily corrected. Moreover, the focus they had was a real problem solution, so this shows that this was more than just a simple investigation.

6.2 Let History not Repeat Itself (this time) — Tackling WebAuthn Developer Issues Early On [34]

- Context

FIDO2 was released recently and they made an analysis of it.

- Problem

A new proposal has arrived the market but not all the problems have been considered.

- Solution

Various points of view in order to cover a wide range of possible problems.

- Comments

The paper includes a lot of references that are supposed to lead to a more detailed analysis.

- Extra details

This article talks about the early approach to FIDO2 in web authentication, as there is no big scale implementation yet. So, the idea is to start investigating existing Web Authentication libraries and specifications in order to identify usability issues that lead eventually to studies for tangible solutions, thus, creating configurations, tool support or "golden guidelines" for Web Authentication developers.

1. The web authentication process is described, how FIDO2 works with web authentication. It points that the RP (Relying party) is responsible for implementing various features.
   The steps are:
   - Registration RP must choose encryption alg., how the user will interact with the service and the authentication attachment (built-in authenticator or external roaming authenticator).
   - Authentication: The user authenticates with the previously registered PK, and the RP must check clone detection of authenticators.
2. WebAuthn pitfalls and issues talks about already existing web Authentication open-source libraries that will be likely used to implement this new method on the top of it.
   - **Mental models:** This was used to understand the people thoughts regarding FIDO2. The thoughts included general and specific questions and people that did not know the real use of FIDO2 authentication model. Finally, they talk about the confusing technical details as the FIDO2 specifications are not simple and developers are confronted with several options and responsibilities.
   - Insecure and incomplete libraries: The top 10 web Authentication libraries on GitHub have problems like they are incomplete, outdated and have poor documentation. This leads to possible security breaches if used.
   - Developer support and education: This talks about what the FIDO alliance is doing to support developers. There are guides and tutorials, but there are a lot of problems like the outdated/non-existent FIDO2 server and so on. There are third party sites that have some articles on this like duo labs, Yubico, Microsoft and google (but most of it is related to specific use-cases their products).
     Conclusion: POOR DOCUMENTATION.
   - Security and privacy concerns: Problems with multiple devices enrollment and secret backdoor access to accounts.

6.3 Keeping Passwords in your Pocket [35]

- Context

Passwords are the primary web authentication mechanism; therefore, the number of passwords people have to manage has increased, generating an increase of similar or recycled passwords for each user, decreasing the security of the web authentication systems.

- Problem

The existent password managers require the memorization of a master password key in order to administrate all the other passwords.

- Solution

Remove the master password of the password managers using finger-print authentication.

- Comments

The way they manage the passwords is not innovative and has a lot of possible problems, starting with the plain text storage of the passwords. Also, the QR codes they generate could be violated by the third party that generates the QR codes, or could be read by another person and thus, another person could obtain the password.

6.4 Research on Integrated Authentication Using Passwordless Authentication Method [36]

- Context:

The Advanced Technology and Science Tokushima University has since 2010 an integrated authentication service using Shibboleth which allows to enter all the services with a pair of ID and password. There they have password policies such as minimum length and combination of some types of characters, also, they must change it periodically.

- Problem

The passwords are problematic. The people tend to forget their complicated passwords or reissued due to expired passwords. Also, having an SSO authentication service, if the pair {ID, password} is leaked, the attacker would have access to all the services. They have tried already to improve security with two-factor authentication, and some important labs have things like ID cards or biometric authentication. However, this are expensive and cannot be implemented full-scale system easily.

- Solution

The authentication system is required to be passwordless, convenient, low-cost and safety. In order to do that, they propose to use FIDO into Shibboleth (university identity provider). Since FIDO identity verification is performed using user's signature, the user's credential information does not flow to the network (preventing MIM credential steal attacks). Also, FIDO allows the users to use their own device (therefore, no extra devices are needed).

- Comments

They give details on the protocol and how the system works. And how to merge it with their university SSO authentication system. Moreover, they consider that even if FIDO solves some problems, there are still things that the developer of the service must be aware of. They also verify the protocol of FIDO with packet tracers and the data flow communication route in order to confirm that no personal data is exposed.

6.5 Should we Rush to Implement Password-less Single Factor FIDO2 based Authentication? [37]

- Context

FIDO2 specifications are relatively new, therefore a lot of platforms have not even considered to implement its SF (single factor) authentication method. The investigation intends to explore the new specifications from various points of view to analyze the weaknesses and strengths of implement FIDO2.

- Problem

The websites are reluctant to adopt password-less FIDO single factor authentication.

- Solution

Compare the actual solutions with the FIDO2 specifications, giving the websites the weaknesses and strengths of both (the first ones are already known). Also, some weaknesses as the usability of FIDO2 are analyzed on this paper, with the intention of stand out the main obstacles of the implementation for the enterprise requirements.

- Comments

This paper has a lot of content, so I decided to make an extended resume highlighting the most important parts. Moreover, I found interesting that they made a very good analysis of the FIDO2 and their usability weaknesses and give some advice on how to affront it.

- Resume

First, they give a description of the FIDO alliance history and their inventions. Mentioning how it has grown and how some of the major web services adopted part of their works. Then they move to a description of the threats to password-based authentication (they made a compilation of various previous studies). With this, they also give a list of the entities that should implement some of the known ways to mitigate these threats. Moreover, some of the already existing solutions to those threats are mentioned.

Then they move and do the same with FIDO. In this case, they compare what FIDO offers and the threats it can cover. Also, they mention that the FIDO system is based on some assumptions to ensure their security goals, as the password-based systems does.

The conclusions show that FIDO SF authentication is more secure than password-based SF authentication. However, FIDO is NOT completely secure but has a smaller attack surface than a password-based system.

Even with this, the companies should not start using FIDO SF authentication because it still lacks important usability features such as:

- Account recovery.
- Account delegation.
- Account deletion/suspension.
- Ease of use.

6.6 SimFIDO – FIDO2 User Authentication with simTPM [38]

- Context:

FIDO2 is a new standard for two-factor and password-less user authentication. It offers various ways to identify like fingerprints, external hardware, etc.

- Problem:

The availability of hardware authenticators that support FIDO2 authentication is focused on desktop computer, while only a few mobile devices can make use of this authentication method. All this along with the classic passwords’ problems.

- Solution:

SimFIDO is a setup of FIDO2 that allows the user to use the FIDO2 authentication protocol. This setup increases the number of devices that can use the authentication method and the moveability as you can easily change the SIM card to another device.

Even though this model is presented, the idea is to develop hardware authenticators more available to the users.

- Comments:

The system is based on TPM. One of the reasons is the mobility that the TEE-based in built-in chips offer.

"However, there are several ways to connect a mobile phone to another platform, e.g., via Bluetooth, NFC or USB. This allows a phone to expose the simTPM as a roaming authenticator to such connected devices." [38]

6.7 No passwords needed: The iterative design of a parent child authentication mechanism [39]

- Context:

The children are getting access to internet and electronic devices at early ages. With this comes the necessity of login in some web sites. Also, the data tells that the number of children that have access to internet is increasing.

- Problem:

Actual authentication mechanisms are not design for children as they imply the usage of passwords. Using a password means that the child must memorize it, and most of the times they forgot the passwords and lock the devices. Also, if they have access to various web sites, the password would be reused. So, the parents end memorizing the password or putting insecure passwords that their children can remember easy.

- Solution:

A multi-platform app was made in order to satisfy the problems identified. This was made in an iterative way that added features on the go, but the main objective (which was the use of an alternative to passwords) was achieved with the use of OPEN ID and FIDO as a second factor authentication.

- Comments:

An important definition is open ID: The OpenID Oauth 2.0 framework and its implementation protocol OpenID Connect support decentralized user-centric authentication, enabling a user to create an account with any OpenID identity provider, such as Google, and use their OpenID credentials to login to any co-operating websites (relying party).

Advantages: password reduction, speedy sing up, interoperability, user control over their online identity.

7 CONCLUSIONS AND FUTURE WORK

Even though the objectives of the project were met, the usability tests should be completed to fully understand the opinion of the users regarding the passwordless environments. Moreover, as the Authenticator is a central component of this kind of environment, a research on the capabilities, strengths and challenges of these devices should be made before fully trust the actual capacity of the passwordless environments using the FIDO2 standard. Finally, the prototype still needs to be tested on different platforms to confirm if other Authenticators like internal authenticators with biometrics do not expose any kind of personal or sensitive information.

In conclusion the deployed environment provides protection to all password related attacks and man in the middle attacks as the personal/sensitive information is encrypted using TLS over HTTP and SAML assertions encryption. However, the system administrator should be aware of the possibility that channels may be compromised.

Furthermore, passwordless environments require a back-up password related or multifactor authentication system defined for highly required services. Otherwise, in the case that a user loses an Authenticator he/she cannot access the desired resources unless the system administrator deletes the actual Authenticator, and the user registers a new one.

Finally, the security when using an external Authenticator relies heavily on the Authenticator vendor as these devices are the center of the authentication process. The user should not use non-certified external Authenticators as those devices could have unknown vulnerabilities.

8 BIBLIOGRAPHY

[1] P. Langlois, "Verizon," 2020. [Online]. Available: https://www.cisecurity.org/wp-content/uploads/2020/07/The-2020-Verizon-Data-Breach-Investigations-Report-DBIR.pdf. [Accessed 28 11 2020].

[2] C. Cimpanu, "ZDNet," 26 11 2020. [Online]. Available: https://www.zdnet.com/article/personal-data-of-16-million-brazilian-covid-19-patients-exposed-online/. [Accessed 30 11 2020].

[3] F. alliance, "Fido overview," [Online]. Available: https://fidoalliance.org/overview/. [Accessed 1 12 2020].

[4] Google/Harris Poll, "The United States of P@ssw0rd$," October 2019. [Online]. Available: https://storage.googleapis.com/gweb-uniblog-publish-prod/documents/PasswordCheckup-HarrisPoll-InfographicFINAL.pdf. [Accessed 5 October 2020].

[5] T. Hunt, "Troy Hunt," 17 January 2019. [Online]. Available: https://www.troyhunt.com/the-773-million-record-collection-1-data-reach/. [Accessed 6 10 2020].

[6] Kaspersky, "Kaspersky," 2020. [Online]. Available: https://www.kaspersky.com/resource-center/definitions/what-is-social-engineering. [Accessed 19 10 2020].

[7] M. Quiroga, "Acerca de biometría," 28 July 2020. [Online]. Available: https://www.linkedin.com/pulse/acerca-de-biometr%C3%ADa-milton-quiroga/. [Accessed 11 October 2020].

[8] F. alliance, "Fido alliance history," 2020. [Online]. Available: https://fidoalliance.org/overview/history/. [Accessed 25 12 2020].

[9] sectigo, Artist, *Public key cryptography diagram.* [Art]. sectigo, 2020.

[10] F. alliance, "What is fido?," 2020. [Online]. Available: https://fidoalliance.org/what-is-fido/. [Accessed 28 12 2020].

[11] F. Alliance, "FIDO UAF Architectural Overview," 02 02 2017. [Online]. Available: https://fidoalliance.org/specs/fido-uaf-v1.1-ps-20170202/fido-uaf-overview-v1.1-ps-20170202.pdf. [Accessed 28 12 2020].

[12] F. Alliance, Artist, *FIDO UAF High-Level Architecture.* [Art]. 2017.

[13] F. Alliance, Artist, *Registration message flow.* [Art]. 2017.

[14] F. Alliance, Artist, *Authentication message flow.* [Art]. 2017.

[15] F. Alliance, Artist, *Deregistration message flow.* [Art]. 2017.

[16] F. Alliance, "FIDO U2F Architectural Overview," 11 04 2017. [Online]. Available: https://fidoalliance.org/specs/fido-u2f-v1.2-ps-20170411/fido-u2f-overview-v1.2-ps-20170411.pdf. [Accessed 28 12 2020].

[17] W3C, "Web Authentication: An API for accessing Public Key Credentials Level 1," 04 03 2019. [Online]. Available: https://www.w3.org/TR/2019/REC-webauthn-1-20190304/. [Accessed 28 12 2020].

[18] FIDO Alliance, "Client to Authenticator Protocol (CTAP)," 30 01 2019. [Online]. Available: https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-client-to-authenticator-protocol-v2.0-ps-20190130.pdf. [Accessed 28 12 2020].

[19] FIDO Alliance, "CTAP - 6. Message Encoding," 30 01 2019. [Online]. Available: https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-client-to-authenticator-protocol-v2.0-ps-20190130.html#message-encoding. [Accessed 28 12 2020].

[20] FIDO Alliance, *Commands structure,* 2019.

[21] FIDO Alliance, *Responses structure,* 2019.

[22] FIDO Alliance, "CTAP - 8.1. USB Human Interface Device (USB HID)," 30 01 2020. [Online]. Available: https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-client-to-authenticator-protocol-v2.0-ps-20190130.html#usb. [Accessed 28 12 2020].

[23] FIDO Alliance, "CTAP - ISO7816, ISO14443 and Near Field Communication (NFC)," 30 01 2020. [Online]. Available: https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-client-to-authenticator-protocol-v2.0-ps-20190130.html#nfc. [Accessed 28 12 2020].

[24] FIDO Alliance, "CTAP - 8.3. Bluetooth Smart / Bluetooth Low Energy Technology," 30 01 2020. [Online]. Available: https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-client-to-authenticator-protocol-v2.0-ps-20190130.html#ble. [Accessed 28 12 2020].

[25] F. Alliance, Artist, *WebAuthn+ CTAP Flow.* [Art].

[26] F. Alliance, Artist, *FIDO Platform/Browser support.* [Art]. 2020.

[27] OpenID, "Welcome to OpenID Connect," 2020. [Online]. Available: https://openid.net/connect/. [Accessed 01 01 2021].

[28] A. Parecki, "What is OAuth and why does it matter? - OAuth in Five Minutes," 2020. [Online]. Available: https://www.youtube.com/watch?v=KT8ybowdyr0&feature=emb_logo. [Accessed 1 1 2021].

[29] J. Petters, "Varonis - What is SAML and How Does it Work?," 29 03 2020. [Online]. Available: https://www.varonis.com/blog/what-is-saml/. [Accessed 31 12 2020].

[30] OASIS Open 2015, "Assertions and Protocols for the OASIS Security Assertion Makup Language (SAML) V2.0 - Errata Composite," 08 09 2015. [Online]. Available: https://www.oasis-open.org/committees/download.php/56776/sstc-saml-core-errata-2.0-wd-07.pdf. [Accessed 01 2021].

[31] OASIS Open, "Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0 - Errata Composite," 08 09 2015. [Online]. Available: https://www.oasis-open.org/committees/download.php/56779/sstc-saml-bindings-errata-2.0-wd-06.pdf. [Accessed 01 2020].

[32] Gluu, Inc., "Gluu server 4.2 Docs - Ubuntu installation," 2020. [Online]. Available: https://gluu.org/docs/gluu-server/4.2/installation-guide/install-ubuntu/. [Accessed 28 12 2020].

[33] D. W. Chadwick, R. Laborde, A. Oglaza, R. Venant, A. S. Wazan and M. Nijjar, "Improved Identity Management with Verifiable Credentials and FIDO," *IEEE Communications Standards,* vol. 3, no. 4, pp. 14-20, 2019.

[34] A. Alam, K. Krombholz and S. Bugiel, "Poster: Let History not Repeat Itself (this Time) -- Tackling WebAuthn Developer Issues Early On," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, London, 2019.

[35] P.-Y. Lin, Z.-y. Zhou, C.-M. Chang, H.-W. Chen, S.-P. Tung and H.-C. Hsiao, "Keeping Passwords In Your Pocket: Managing Password Locally With Mobile Fingerprint Sensors," in *Companion Proceeding of the Web Conference 2020*, New York, 2020.

[36] M. Morii, H. Tanioka, K. Ohira, M. Sano, Y. Seki, K. Matsuura and T. Ueta, "Research on Integrated Authentication Using Passwordless Authentication Method," in *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, Turin, 2017.

[37] F. Alqubaisi, A. S. Wazan, L. Ahmad and D. W. Chadwick, "Should We Rush to Implement Password-less Single Factor FIDO2 based Authentication?," in *2020 12th Annual Undergraduate Research Conference on Applied Computing (URC)*, Dubai, 2020.

[38] D. Chakraborty and S. Bugiel, "SimFIDO: FIDO2 User Authentication with simTPM," in *Conference on Computer and Communications Security*, New York, 2019.

[39] K. Hundlani, S. Chiasson and L. Hamid, "No passwords needed: the iterative design of a parent-child authentication mechanism," in *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services*, New York, 2017.

[40] FIDO Alliance, "CTAP - 8. Transport-specific Bindings," 30 01 2019. [Online]. Available: https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-client-to-authenticator-protocol-v2.0-ps-20190130.html#transport-specific-bindings. [Accessed 28 12 2020].

[41] A. Parecki, "OAuth 2.0," 2020. [Online]. Available: https://oauth.net/2/. [Accessed 31 12 2020].