Digital Security: Cybersecurity, Privacy and Trust
DS-02-2014 Access Control

European Commission | Horizon 2020 European Union funding for Research & Innovation

From Real-world Identities to Privacy-preserving and Attribute-based CREDentials for Device-centric Access Control

WP2 – Requirements, Business Cases and Architecture

Deliverable D2.7 "Reference architecture - revised"

Editor(s):	Evangelos Kotsifakos (WEDIA)
Author(s):	Savvas Zannettou (CUT), Charalambos Chrysostomou (CUT), Kostantinos Papadamou (CUT), Michael Sirivianos (CUT), Michalis Pramateutakis (WEDIA), Giorgos Gonidelis (WEDIA), Alberto Caponi (CNIT), Claudio Pisa (CNIT), Giuseppe Bianchi (CNIT), George Gugulea (CSGN) Bogdan Chifor (CSGN), Ionut Florea (CSGN), Valentin Necoara (CSGN), Steven Gevers (VER), Arsalan Najwani (VER)
Dissemination Level:	PU - Public
Nature:	R
Version:	0.7

ReCRED Project Profile

Contract Number	653417
Acronym	ReCRED
Title	From Real-world Identities to Privacy-preserving and Attribute-based CREDentials for Device-centric Access Control
Start Date	May 1st, 2015
Duration	36 Months

Partners

University of Piraeus	University of Piraeus research center	Greece
Telefónica Investigación y Desarrollo	Telefonica Investigacion Y Desarrollo Sa	Spain
verizon	Verizon Nederland B.V.	The Netherlands
certSIGN BY UTI	Certsign SA	Romania
wedia	Wedia Limited	Greece
EXUS	EXUS Software Ltd	U.K.
upcom Bringing business and IT together	Upcom Bvba (sme)	Belgium
THE PRODUCTIZERS	De Productizers B.V.	The Netherlands
2004	Cyprus University of Technology	Cyprus
	Universidad Carlos III de Madrid	Spain
cnit	Consorzio Nazionale Interuniversitario per le Telecomunicazioni	Italy
BAKER & MCKENZIE	Studio Professionale Associato a Baker & Mckenzie	Italy

Document History

Version	Date	Author	Remarks
0.10	2016-11-14	WEDIA	Distribution of first draft/ instructions to partners
0.2	28-11-2016	Charalambos Chrysostomou (CUT), Savvas Zannettou (CUT)	Deliverable restructuring and update. Revision of SP, BAA and ID providers.
0.3	06-12-2016	Claudio Soriente (TID), Christoforos Dadoyan (UPRC), Ozlem Bilgili (Verizon), Evangelos Kotsifakos (WEDIA), Savvas Zannettou (CUT)	Input from TID, UPRC, Verizon, CUT
0.4	27-12-2016	Evangelos Kotsifakos (WEDIA), Savvas Zannettou (CUT) , George Gugulea (CSGN), Alberto Caponi (CNIT), Claudio Pisa (CNIT), Giuseppe Bianchi (CNIT), Michalis Prama-teutakis (WEDIA), Michael Sirivianos (CUT)	Input from CNIT, CSGN, CUT, WEDIA
0.5	28-12-2016	Evangelos Kotsifakos (WEDIA), Savvas Zannettou (CUT)	Integration of input – first review
0.6	20-06-2016	Kostantinos Papadamou (CUT)	Added the threat and vulnerability risk assessment table
0.7	27-07-2017	Kostantinos Papadamou (CUT)	Updated threat model

Executive Summary

This document is part of the WP2- *Requirements, Business Cases and Architecture* of the ReCRED project. The purpose of this report is to define and describe the various components of the ReCRED framework architecture, the interaction between them and the technologies that are used for each component. In order to clearly highlight the requirements that led to this architecture a complete threat model is used. The breakdown of the various components has also been based on horizontal use cases and user stories that are part of the deliverables D2.5 and D2.6. As this is the revised version of the deliverable D2.3 it contains the finalized architecture that is the basis of the project implementation. The ReCRED framework consists from the following components: the Identity Consolidator which plays a central role, the Service Providers, the Identity Providers, the Behavioral Authentication Authorities and the User’s Device. Apart from the detailed description of these components, this report also describes the state-of-the-art protocols that are used to connect each-other and support security and privacy: FIDO, OpenAM, Mobile Connect, Idemix and U-prove, gateSAFE and XACML.

The reference architecture is closely related to the functional and technical requirements of the ReCRED platform. The connection between the architecture and the actual implementation of the platform can be found in D2.6 where the requirements per architectural component are analysed and linked to tasks of the Phabricator tool that is used to track and organize the project’s implementation.

Table of Contents

Executive Summary 4
List of Figures 7
Table of Abbreviations 9
1 Introduction 10
1.1 Threat model 10
2 Architecture Components 17
2.1 User’s Device 20
2.2 Identity consolidator 23
2.2.1 Physical Identity Acquisition Module 26
2.2.2 Online Identity Acquisition Module 29
2.2.3 Account Management Module 31
2.2.4 Credential Management Module 35
2.2.5 Authentication Management Module 38
2.2.6 Identity Management Module 39
2.2.7 Storage API and Repository schema 42
2.2.8 3rd Party API 43
2.3 Service Providers 44
2.3.1 OpenID Connect Relying Party (Federated Logins) 45
2.3.2 Access Control Policy Reasoning Tool 46
2.4 Behavioral Authentication Authorities 47
2.4.1 Behavioral profiles database 48
2.4.2 Behavioral profiles extraction 49
2.5 Identity Providers 49
2.5.1 Identity Providers FIDO UAF Server 50
2.5.2 QR Authentication Server 51
2.6 Privacy-Preserving Attribute Based Access Control 53
3 Protocols and Interfaces 55
3.1 FIDO Integration in the ReCRED Architecture 55

3.1.1 Registration Protocol 58
3.1.2 Authentication Protocol 58
3.1.3 Federated Authentication Protocol 59
3.1.4 De-registration Protocol 60
3.1.5 FIDO UAF Server 61
3.2 OpenID Connect/OAuth 66
3.2.1 Introduction 66
3.2.2 OpenID Connect 66
3.2.3 Protocol Overview of OpenID Connect 68
3.2.4 OpenAM 69
3.3 Mobile connect 70
3.4 Cryptographic Credential Stacks 73
3.4.1 Idemix 73
3.4.2 U-prove 79
3.5 ReCRED integration 85
3.5.1 FIWARE interfaces 86
3.6 gateSAFE 87
3.6.2 gateSAFE in ReCRED 91
3.7 XACML 93
3.7.1 Introduction 93
3.7.2 XACML Architecture 94
3.7.3 XACML and ReCRED 95
4 Conclusions 96
5 References 97

List of Figures

Figure 1 ReCRED Reference Architecture 10
Figure 2 Device Centric Authentication Component View 18
Figure 3 User's Device Protocol Stack View 20
Figure 4 Identity consolidator protocol stack view 24
Figure 5: Identity consolidation with identity acquisition and verification component view 26
Figure 6 Physical Identity Acquisition and verification process 27
Figure 7 Online Identity Acquisition Module 29
Figure 8 Architecture and Interfaces for the Credential Issuance service provided by the Credential Management Module 36
Figure 9 Storage Model 43
Figure 10 Service Providers protocol stack view 44
Figure 11 Behavioral authentication authorities protocol stack view 47
Figure 12 Identity Provider protocol stack view 49
Figure 13 QR Authentication architecture 52
Figure 14: Attribute-Based Access-Control component view 55
Figure 15 FIDO Components 56
Figure 16 FIDO Authentication Protocol 59
Figure 17 FIDO and Federated Authentication Protocols 60
Figure 18 SAML 2.0 & OpenID Connect comparison 68
Figure 19: OpenID Connect protocol execution flow 69
Figure 20 Mapping of Identity Provider and User Device respectively to an Issuer and a Receiver running the Issuing Protocol to allow the release of credential 73
Figure 21 Mapping of Online Service and User Device respectively to a Verifier and a Prover running the Proving Protocol to allow the proof of possession of credentials 73
Figure 22 IRMA Architecture 77
Figure 23 IRMA verification protocol 77
Figure 24 IRMA Issuance Protocol 78
Figure 25. U-Prove Roles 79
Figure 26. U-Prove token public components. 81
Figure 27. U-Prove token private components. 81
Figure 28. Parties and components involved in the Issuance Protocol. 82
Figure 29. Parties and components involved in the Presentation Protocol. 82
Figure 30 – U-Prove token issuance 84
Figure 31 – U-Prove client-server architecture 85
Figure 32 ReCRED PABAC architecture with FIWARE APIs 86
Figure 33 Usual architecture where users have a user and password for each web application 87
Figure 34 gateSAFE makes it easier to manage user identities by using Digital Certificates 88
Figure 35 XACML Architecture (Source: Wikipedia) 95

Table of Abbreviations

ABAC	Attribute-Based Access Control
ASM	Authenticator Specific Module
BAA	Behavioral Authentication Authority
CCS	Cryptographic Credentials Storage
DM	Delegation Mediating
DCA	Device-Centric Authentication
FIDO	Fast IDentity Online
FIDO UAF	FIDO Universal Authentication Framework
IdP	Identity Provider
JOSE	JSON Object Signing and Encryption
MTM	Mobile Trusted Module
PoP	Proof of Possession
QSEE	Qualcomm Secure Execution Environment
SE	Secure Element
SAML	Security Assertion Markup Language
SP	Service Provider
SSO	Single-Sign-On
SoC	Systems on a Chip
TEE	Trusted Execution Environment
TPM	Trusted Platform Module
XACML	eXtensible Access Control Markup Language

1 Introduction

The objective of the ReCRED project is to implement an architecture that facilitates seamless integration of the multiple identity attributes of a user, physical and online, under a secure identity consolidator. The Device-Centric Authentication (DCA) is central to our architecture and the end-user can access the services only through his device. Our DCA components go beyond the FIDO (Fast IDentity Online) standard [11] by addressing its weaknesses. Furthermore, we surround DCA with additional components that address identity fragmentation, provide real-world binding, address privacy challenges and provide the ability for complex access control policy reasoning. The following figure presents the architecture building blocks and the flow between them.

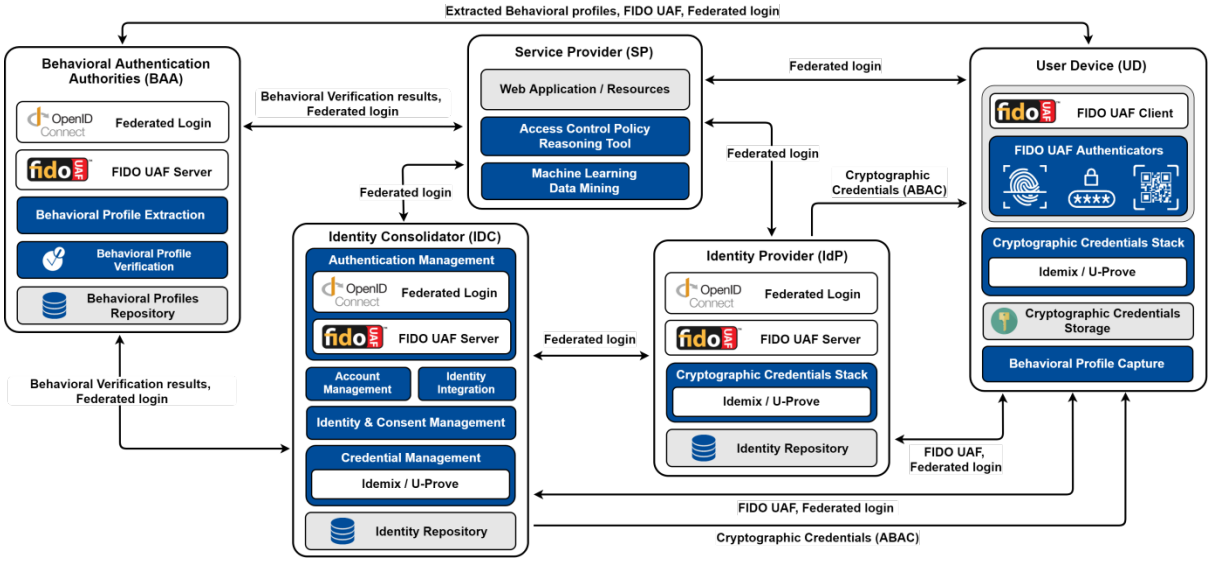


Figure 1 ReCRED Reference Architecture

These building blocks are analysed in the next chapters. Protocols and technologies that will be used to implement this architecture are described in Section 3.

The functional and technical requirements that are related to each one of the components are analysed in Section 11 of the deliverable D2.6.

The detailed specifications of the protocols and the components of the architecture are also described in detail in the corresponding work package deliverables.

1.1 Threat model

In order to better understand and define the requirements for the ReCRED final reference architecture, we describe the following threat model.

For the user device, we distinguish among the following attacker capabilities:

1. The attacker can steal the device but cannot perform S/W attacks. ReCRED is able to effectively prevent these kinds of threats by employing the thrust of the Behavioral Authentication Authorities. Specifically, the diverse behavioral multi-factor authentication can prevent an attacker from being authenticated as the legitimate user. (UDT.1)
2. The attacker steals the device and can perform S/W attacks:
    a) If the protocols run in the Normal World, a skilled S/W attacker can change the memory contents and intercept or modify information from sensors. If the behavioral factor derives from a local measurement (e.g., gestures, acceleration, etc.) the attacker can read the prior measurements and replicate them so that the device appears to be held by its legitimate user. To protect against this case, all local measurements have to be sent immediately to the Behavioral Authentication Authority (BAA). In this case, the attacker cannot bypass the behavioral authentication, because he does not know the signature (under specific temporal assumptions regarding the behavior capturing, transfer and erasure). We assume that the behaviors have high entropy and thus can be hard to guess unique identifiers. If the behavioral factor derives from remote measurements (e.g., analysis of traffic pattern or mobility behavior done by the Mobile Operator, who acts as a BAA), the attacker who stole the device cannot authenticate as the legitimate user. The behavioral profiles (signatures) are cryptographically signed with by the ReCRED credential that verifies, they are bound to the device and originate from the legitimate user. (UDT.2a)
    b) If the protocols run in the Secure World (aka Trust-Zone, or Trusted Execution Environment (TEE)), no S/W attacker can compromise the memory and information paths. However, we do not have the ability to develop protocols for TEE as the trusted computing base has to be approved by vendors such as Intel and/or Samsung, etc. We can just invoke specific services of it, such as storing keys and performing secure cryptographic operations. For example, the activation of behavioral authentication with a verifier can be triggered by TEE-enabled secure biometrics (fingerprint). This last measure is supplied by FIDO and can protect the user in case the device has been recently stolen, and the behavioral signature has yet to change. (UDT.2b)
3. We assume that the MTM (Mobile Trusted Module) does not have a trusted path to the sensor, so there is no point having the MTM signing the behavioral profiles. If the opposite held, then MTM-signed profiles would be secure against a hacker. No such functionality is available on commodity handheld devices. (UDT.3)
4. The attacker steals the device and can perform H/W plus S/W attacks. Behavioral authentication, as described above (local with secure erasure and transfer to BAA or remote), protects the user in this scenario. But the attacker can bypass the trusted authentication activation and present himself as the legitimate user if the device is recently stolen. (UDT.4)
5. The attacker does not steal the device, but remotely performs a software attack. He can bypass the behavioral authentication because the device is still held by its legitimate user. If

the authentication is triggered securely (e.g., FIDO biometric), he cannot initiate it by himself. But he still can wait for the user to do it. If the verifier challenges the user to authenticate, and the system requires the user to trigger the authentication through a TEE, the hacker cannot authenticate on-demand. With a TEE initiating the authentication, we raise the bar for the attacker because he cannot login on demand. (UDT.5)

For the ReCRED online services, we distinguish among the following threats:

1. Access Token Disclosure and Server Response Disclosure: The ReCRED platform will guarantee that the authentication tokens will never be exposed to unauthorized parties, as they represent the users' authentication. Additionally, the server response can contain authentication and sensitive user information. All server responses will be sent over TLS protected channels. (OST.1)
2. Cross Site Request Forgery (CSRF): where the attacker forces the user to execute unwanted actions to services, they are authenticated and have access. ReCRED will make use of best practices for preventing these kind of threats. Specifically, we will perform header checks in order to verify the origin of the source and destination. Furthermore, we will make use of CSRF tokens. (OST.2)
3. Man-in-the-Middle Attack (MitM): where the attacker intercepts a communication between two systems for various malicious purposes. Examples:
    a. Token Manufacture/Modification: An attacker can modify the content of an existing token or generate a fake token that will provide inappropriate access to a Service Provider. The ReCRED platform will ensure that tokens will be digitally signed by the original poster, and that the Relying Party will validate the digital signature. Additionally, the tokens will always be exchanged over protected TLS channels. (OST.3a)
    b. Access Token Redirect and Token Reuse: An Attacker can obtain and use the Access Token generated by the Identity Provider in order to obtain additional access to resources. Additionally, the attacker can reuse one-time tokens like an Authorization Code that has already been used and gain access to the intended resources. In order to deal with this type of threats the ReCRED platform will generate Access Tokens that are user and scope restricted. Additionally, the issued tokens will include a timestamp and a short expiration time. (OST.3b)
    c. Token Substitution: The attacker can change various tokens, including substituting an Authorization Code of a user with another token that the attacker has. Furthermore, the attacker may attempt to impersonate a more privileged user, like a system administrator by corrupting the communication channel between the server and the Client. Additional mechanisms will be employed by the ReCRED platform to address this issue. One such mechanism would be to include an ID Token in the request and response tokens. (OST.3c)
4. Pharming Attack: The attacker corrupts an infrastructure service causing the user to be redirected to a fake service provider. This attack can cause the user to reveal sensitive information. An example is the Server Masquerading attack where an attacker can use a

malicious server and masquerade as the legitimate server using various means. By using the ReCRED platform to provide the trusted identity to Service Providers such an attack can be easily detected. Additionally, the user device will authenticate against the server of a trusted Identity Provider, using Signed and Encrypted JWTs with an appropriate key and cipher. (OST.4)

5. TLS Attacks: using a specialized component to solve a very specific task in ReCRED: communication encryption and messages authentication, we ensure that specific attacks that are performed can be avoided; e.g. compression attacks (CRIME, TIME, BREACH), renegotiation attack, private keys theft, weak cryptographic algorithms, to name just a few of the attacks that happened over time, deploying gateSAFE in the ReCRED architecture ensures a higher grade of security is obtained. (OST.5)
6. Leak of information from Service Providers: This is a case where a malicious Service Provider leaks information of users that were presented using the OpenID Connect specification. The ReCRED architecture ensures that the leaked information cannot be used for Authorization and Authentication of users to other online services. In other words, an attacker can not impersonate a legitimate user from the leaked information to other Identity Providers or Service Providers. (OST.6)
7. Replay Attacks: This is the case when an eavesdropper intercepts a successful authentication of a legitimate user with an Identity Provider before redirection to the Service Provider. ReCRED prevents these kinds of replay attacks by using transport layer encryption for the connections between the user device and the Identity Providers/Service Providers.

The ID Consolidator is vulnerable to the following threats:

An important threat that we have to defeat against in ReCRED is the scenario where an attacker steals the device of a user and tries to access the ID Consolidator and all the user’s accounts. In order for an attacker to use the stolen device he has to first bypass local authentication if the user has set one. If the attacker is not smart enough to bypass local authentication then we are protected. In the case that the user is smart enough to bypass local authentication then he is able to use the stolen device. At this point it is likely that the attacker will use the device in a different way that the legitimate user does and their behaviors on the device do not match. This is the reason we employ continuous behavioral authentication in order to detect unusual behavior on a device. Continuous behavioral authentication happens remotely on the BAAs. We chose to store the behavioral signatures remotely on the BAAs instead of having them stored locally in the device because a good attacker can steal and use them in his device to impersonate the legitimate user. When a continuous authentication has failed, which means that the BAA detected that the holder of a specific device is not behaving and usual, then the BAA calls Account Management Module (AMM) and latches all the accounts of the user on the specific device. In this case the attacker is not able to access any of the user’s account on the stolen device and the user is protected. At the same time the attacker is not able to access the IDC from the stolen device. (IDC.1)

Now assuming that the attacker is smart enough to bypass continuous authentication then the user can protect himself by declaring his lost device as stolen. This can be done by logging into the IDC

using his master password from a new device or his desktop PC and then he is able to declare the device as stolen through the AMM and lock all his account on the specific device. At the same time the FIDO key installed on the stolen device gets invalidate and the attacker is not able to perform FIDO authentication anymore using the stolen device. However, we admit that there is a vulnerability window from the time that the device has been stolen to the time when the user will declare this device as stolen. In this case we rely on the FIDO human-to-device authentication to protect and prevent unauthorized access to the IDC or the accounts of the user. (IDC.2)

Additionally, even if the legitimate user has invalidated the FIDO key of the stolen device, the attacker could have guessed or know the user’s IDC master password and may try to use the stolen SIM card to perform failover authentication with Mobile Connect and get full access to the user’s IDC account in case the user has not declared it lost. We are able to defeat against such attack by also requiring failover authentication using a BAA in order to verify the behavior of the user before granting him access to his IDC account. (IDC.3)

Furthermore, even if the legitimate user was able to declare his device as stolen and the FIDO key got invalidate before the attacker tries to use the stolen device, the attacker could have guessed or know the user’s IDC master password. In case the attacker tries to login to the IDC using the master password of the user he will be granted with only tentative access to the IDC. This means that the user has access to limited functionalities since no FIDO authentication has been established so far and the IDC does not yet trust him. In particular, the attacker cannot view or restore the credentials of user, and he cannot view the identity attributes of the user. He can only see which IdPs and BAAs the user has. At this time we can claim that we are safe since the attacker has to perform failover authentication either using Mobile Connect or a BAA in order to get full access to the user’s IDC account. As soon as the user has declared the device as stolen, the SIM card has been invalidated and the attacker cannot use Mobile Connect. If the attacker tries to perform failover authentication using a BAA he has to know the user’s BAA backup password. Even if we assume that he knows the backup password he will not be able to bypass the behavioral authentication since his behavior will not the user’s old behavior on this device. (IDC.4)

Physical Identity Acquisition and NFC ID Document verification threats:

1. The attacker steals the ID document (Identity card or passport) of a user and tries to verify it using the Physical identity acquisition module claiming that it is his: If the attacker tries to verify a forged or stolen identity document following the standard physical identity acquisition process which involves capturing photos of the document we are able to detect and prevent such malicious activity. We will detect this when we are requesting from him to capture photos of his face because these photos will not match the photo of the user in the identity document photo. We also exploit the Trusted Execution Environment of the user device and by employing cryptographic techniques we ensure the authenticity and fidelity of the captured photos which allow us to ensure that the submitted photos have been captured from the camera of the user’s device. (PIA.1)

2. An attacker steals the ePassport of someone else and tries to verify that it is his using the NFC ID Document verification process offered by the Physical Identity Acquisition module. In this case we cannot do anything to prevent this because the server will identify that the personal data sent to the server came from a genuine ePassport. (PIA.2)
3. An attacker clones the ePassport of another user and tries to verify his identity using NFC ID Document verification process claiming that he is someone else. We are able to defend against such an attack be verifying whether the data read from an RFID chip have been read from the genuine ePassport and not from a copied or cloned one. This is achieved with Active Authentication to verify the authenticity of the ePassport. Active Authentication uses a challenge response mechanism to verify whether the data read from an RFID chip of an ePassport belongs to the genuine document. The requirements for remote ePassport verification are much more than the standard local ePassport verification. In the case of remote ePassport verification the server needs to ensure that the received data have belongs to the genuine document. In order to achieve this we are performing Active Authentication between the server and the ePassport (the server sends the challenge) with the user device (ePassport reader) being just a proxy that scans the data from the ePassport. (PIA.3)
4. Impersonation with a replay attack: An attacker may tries to read the ePassport data that another user tries to send to the server by performing a replay attack. By having the server performing Active Authentication as we do for PIA.3 above we are able to detect and prevent such type of attacks. (PIA.4)

User privacy threats:

One of the most important factors in the acceptance and use of web services is the protection of users’ privacy. Service Providers may require authentication or accountability of users’ actions, in which case users need to prove their identity, or at least possession of a certificate or capability of a certain type. These Service Providers can identify a user through a combination of context from a series of transactions. Moreover, even if standard anonymization practices are performed by the user, if the issuer and the verifier are in communication, the user can be identified directly. ReCRED will preserve end-users privacy by employing advanced anonymization and unlinkability cryptographic protocols, such as U-Prove and Idemix. The multi-show unlinkability will be guaranteed for Idemix, and the untraceability guarantee will be preserved for U-Prove. (UP.1)

Furthermore, the user will be able to give his consent when revealing identity attributes to Service Providers. This will be achieved using the consent management module within the Identity Consolidator. Also, by using the OpenID Connect specification, the user can accept the release of his identity attribute, thus giving explicit consent for the reveal of his attributes. These two mechanisms will allow the user to maintain his privacy if he desires.

Other Security Threats:

1. Denial of Service (DoS): is a technique used by attackers to effectively shut-off access to sites and services. The attackers can accomplish this attack by increasing traffic on the site or

service in a degree that it becomes unresponsive. ReCRED will utilize upstream filtering; all traffic that is passed to the service will be filtered via various methods such as proxies, tunnels or even direct circuits, and the "bad" traffic will be filtered and only the legitimate traffic will reach the service. (OT.1)

2. Cross-site Scripting and SQL Injection: Cross-site scripting or XSS is a technique that makes use of vulnerabilities in web applications. This vulnerability allows the attacker to inject code in a server-side script that they will use to execute malicious client-side scripts or gather sensitive data from the user. The SQL Injection requires a vulnerability to be present in the database associated with a web application. The malicious code is inserted into strings that are later passed to the SQL server, parsed, and executed. Both Cross-site Scripting and SQL Injection vulnerabilities can be prevented by scanning for code and fixing any identified issues. (OT.2)
3. The ReCRED project will deploy the best practices for securing the online service from all common threads, including encryption of data, usage of digital certificates, implementation of data loss prevention and auditing plan, implementation of a removable media policy, securing services against man-in-the middle and malware infections, using network-based security hardware and software, maintain security patches and finally educating the users. (OT.3)

Threat Code	Threat Description	Threat Probability	Vulnerability Probability
UDT.1	The attacker can steal the device but cannot perform S/W attacks	High	Low
UDT.2a	The attacker steals the device, can perform S/W attacks and the protocols run in the Normal World	Medium	Low
UDT.2b	The attacker steals the device, can perform S/W attacks and the protocols run in the Secure World	Medium	Medium
UDT.3	The attacker steals the behavioral profiles of the user as they are generated from the sensors	Low	Low
UDT.4	The attacker steals the device and can perform H/W plus S/W attacks	Low	Medium
UDT.5	The attacker does not steal the device, but remotely performs a S/W attack	Medium	Low
OST.1	Access Token Disclosure and Server Response Disclosure	Low	Low
OST.2	Cross Site Request Forgery (CSRF)	High	Low
OST.3a	Man-in-the-Middle Attack (Token Manufacture/Modification)	High	Low
OST.3b	Man-in-the-Middle Attack	High	Low

	(Access Token Redirect and Token Reuse)		
OST.3c	Man-in-the-Middle Attack (Token Substitution)	High	Low
OST.4	TLS attacks	High	Low
OST.5	Leak of information from Service Providers	High	Low
OST.6	Replay attacks	Medium	Low
IDC.1	The attacker steals the device and tries to access the IDC	High	Low
IDC.2	The attacker steals the device, bypasses continuous authentication and tries to access IDC	Low	Medium
IDC.3	The attacker tries to perform failover authentication with Mobile Connect using the stolen SIM car	Medium	Low
IDC.4	The attacker knows the IDC master password of the user and tries to access the user’s IDC account	High	Low
PIA.1	The attacker steals the ID document (Identity card or passport) of a user and tries to verify it using the Physical identity acquisition module claiming that it is his	Low	Low
PIA.2	The attacker steals the ePassport of someone else and tries to verify that it is his using the NFC ID Document verification process	Low	High
PIA.3	An attacker clones the ePassport of another user and tries to verify his identity using NFC ID Document verification process claiming that he is someone else	High	Low
PIA.4	Impersonation with a replay attack	Low	Low
UP.1	User de-anonymization	High	Low
OT.1	Denial of Service attacks	High	Low
OT.2	Cross-site Scripting and SQL Injection	High	Low
OT.3	Man-in-the middle and malware infections	High	Low

2 Architecture Components

In this section, we describe the reference architecture of the ReCRED platform focusing on DCA which is its central functionality. In order to operate properly, DCA needs multiple sub-components

to interact with each other. Each sub-component consists of multiple modules, as shown in the diagram below. We provide a detailed description for each sub-component, its modules, and how it interacts with the others sub-components.

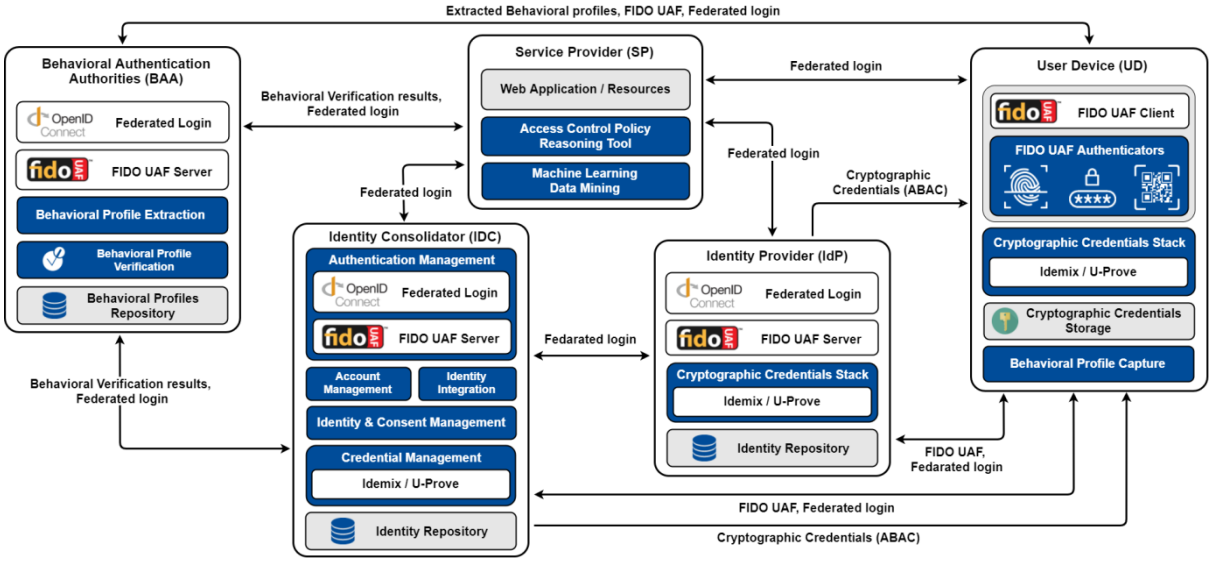


Figure 2 Device Centric Authentication Component View

The reference architecture components are the following:

1. User Device
2. Identity Consolidator
3. Service Providers
4. Behavioral Authentication Authorities
5. Identity Providers

A special reference is made for the Attribute-Based Access Control method.

The ReCRED architecture is based on the following main pillars:

- The ReCRED architecture proposes the use of a component, which we call Identity Consolidator, that will act as the main gateway for proving attributes to third-parties. User’s of the ReCRED platform will have a wide-variety of features on the Identity Consolidator that will help them improve their online experience, security and privacy. More information about the Identity Consolidator and all of its features can be found in Section 2.2.
- The ReCRED architecture is built on top of the OpenID Connect specification. All the necessary information that need to be exchanged between the various components will make use of the OpenID Connect specification.

- Service Providers should require none-to-minimal modifications in order to use the ReCRED platform. As can be seen by Figure 2, the Service Providers will only communicate with other entities using Federated Logins specifications (e.g., OpenID Connect) and will incorporate their business logic through their access control policies. More information about the Service Providers entity can be found in Section 2.3.
- The ReCRED platform aims to offer a delegated behavioral authentication solution. In order to achieve this, we propose the use of a distinct entity with the whole purpose of providing behavioral authentication (both on-demand and continuous). To this end, the ReCRED architecture proposes the Behavioral Authentication Authorities (BAAs) that track users’ behavior and offer a behavioral solution to either the Service Providers or to the Identity Consolidator. The outcome of behavioral authentication will be released to the aforementioned entities using the OpenID Connect specification. We will treat the outcomes as “identity attributes” that will be revealed by an Identity Provider (in this case the BAAs). More information about the BAAs can be found in Section 2.4.
- The ReCRED architecture is based on a set of platforms and specifications that are very popular and greatly used in today’s Internet. Such specifications include the OpenID Connect, the FIDO UAF and the XACML standard. Furthermore, the ReCRED architecture encompasses innovative solutions, that derived from research work, for cryptographic credentials such as Idemix and U-Prove. More information about the used protocols/interfaces and specifications can be found in Section 3.
- Within the ReCRED architecture we aim to provide a diverse authentication framework that will offer a variety of level of assurances. For example, we will use hard cryptographic keys for the authentication of end-users with the highest degree of assurance (LoA4). In order to achieve this, we will use an enhanced FIDO UAF specification that takes into advantage Trusted Execution Environments (TEE) that run on end-user devices. For more information about the supported authentication methods see Section 2.2.5.
- The ReCRED architecture will offer, though its Identity Consolidator, Mobile Connect as a Service. To achieve this, the Identity Consolidator will be the gateway for discovering Mobile Network Operators (MNO) for end-users by contacting the APIGEE API (maintains MNO to mobile phone number mappings). To this end, the Service Providers should not be aware of the APIGEE API. Furthermore, in this context we will make use of the Mobile Network Operators as any other IdP within the ReCRED architecture by using the OpenID Connect specification. For more information see Section 3.3.
- The ReCRED architecture will offer a Federated Privacy-Preserving Attribute-based Access Control solution (PABAC). To achieve this, we integrate the Idemix and U-Prove stacks within the OpenID Connect Provider implementation (OpenAM) on the Identity Providers. This solution has various advantages which are:
    - The PABAC deployment is much simpler for Service Providers because they don’t have to deploy cryptographic stacks on their architecture. Instead, they delegate the verification of the cryptographic credentials to Identity Providers and they receive the verified identity attributes through the OpenID Connect specification.

- This solution allows for more flexibility as PABAC-enabled ID providers might not be collocated with Service Providers

More details about the PABAC solution can be found in Section 2.6.

In the next sections we describe the various ReCRED components and the architectural decisions that we made for each component.

2.1 User’s Device

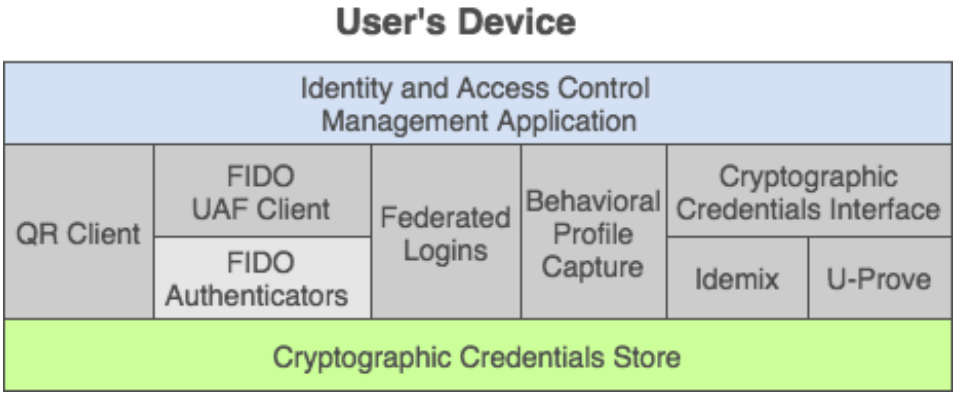


Figure 3 User's Device Protocol Stack View

The mobile device of the user is an important ReCRED component as we aim to achieve a device-centric authentication. During an authentication with a remote service, the user is prompted for Human-to-Device local authentication mainly using biometrics. This is achieved with the multi-factor authentication mechanisms module, which extends the current FIDO UAF (Universal Authentication Framework) protocol [12]. This module has a **FIDO UAF client** with a FIDO UAF protocol stack installed. This module also supports all the types of human-to-device authentication that FIDO supports (such as biometrics, behavioral signatures, and short pins). Furthermore, the user’s device interacts with the Identity Providers for authentication purposes. ReCRED can augment FIDO to enable Attribute-Based Access Control (ABAC) cryptographic credentials [2] to Identity Providers.

On the user’s device we deploy a variety of protocols. Specifically, the user device has a **behavioral profile capturing module**, which acquires behavioral attributes of the user, such as typing pattern, gait etc. Those captured behavioral profiles are transferred securely and stored to the BAAs (Telcos, WEDIA, etc.). Moreover, the user’s device has a **cryptographic credentials storage**, which allows the device to store the cryptographic credentials it receives from the ID Consolidator and from multiple ID Providers. The cryptographic credentials storage takes into advantage the TEE capabilities of the user device to ensure that the credentials cannot be tampered.

The device also runs **Federated Logins protocols** (OpenID Connect/OAuth) with the ID providers and Service Providers (aka, relying parties) for authorization and authentication purposes. To augment OpenID [25], we replace the password with FIDO UAF authentication between the device and the Identity Provider.

On the user’s device there is also an **identity management application**, which communicates with the ID consolidation service to allow users to manage their identity information. With this application, a user can do various things. First, he can issue cryptographic credentials from his identity attributes directly to his device and use them for ABAC. The application running on the device is responsible to store those credentials to the cryptographic credential storage. Second, he can create verifiable partial profiles that contain his desired identity attributes. This application also empowers users to maintain control over the identity attributes revealed to each Service Provider. The identity management application also allows the user to configure his device and his ID consolidator account with a list of his Identity Providers. In addition, the identity management application incorporates consent management and an ID privacy functionality that enables users to have knowledge and control over which Service Provider know which aspects of his identity. For example, it informs the user that an age-restricted service requires verification of his age and whether that service has acquired knowledge of his age through an authentication process. It also allows him to transfer ID attributes among ID providers and between the ID providers and the ID consolidator.

The user device also incorporates a module that is used for transferring credentials using QR codes (**QR client**). This used for creating a new session for a user that already authenticated. For example, when a user authenticates using his mobile device and then he wants to access the Service from his browser, then he is prompted to scan a QR code which is used by the Identity Provider in order to verify that the user is authenticated and that a valid session is on-going in order to create a new session or transfer and already existing session (to a new browser or device) so that access can be granted to the user without the need of re-authentication.

The user device also includes an implementation of a **cryptographic interface** that is used to communicate with the **Idemix** and **U-Prove** stacks that are deployed on the user’s device. These stacks are responsible for releasing Idemix and U-Prove credentials that are stored within the Cryptographic Credentials Storage.

FIDO stack on user's device

The FIDO UAF Client implements the client side of FIDO and transports the underlying protocol specific authentication and registration messages. It is responsible for:

- Communicating with the FIDO server, not directly but through a user agent (an application that runs on the user device that can be either a mobile application or a standard internet browser). Therefore, the FIDO client will interface with the user agent and will deliver and receive messages from it. The user agent forwards FIDO protocol specific messages to the FIDO server through the relying party web application. The user agent contains a FIDO-specific implementation (e.g. a browser plugin) that handles the FIDO messages.

- Communicating with the FIDO UAF Authenticator, exchanging protocol specific cryptographic messages. The FIDO Client communicates with the FIDO Authenticator using a standardized interface: the Authenticator Specific Module (ASM), which provides a uniform interface between the client and the hardware.

The ASM is an abstraction layer that provides a uniform API between the authenticator hardware and the FIDO Client. The FIDO Client uses the ASM API calls in order to delegate cryptographic operations to the Authenticator module. Also, the ASM interface allows the use of different authenticator modules and ensures the compatibility between the authenticator and the client.

The FIDO UAF Authenticator is the secure element connected to all or part of the user device (mobile phone) that is responsible for the creation of the cryptographic key material, storage and secure usage of the private keys used in the authentication protocol. In the case the device contains and uses a Trusted Execution Environment (e.g., ARM Trust Zone), the authenticator can use this safe storage and execution memory/CPU to obtain a higher level of security.

Cryptographic Credentials Storage

The user’s device has a Cryptographic Credentials Storage (CCS) where the device stores in a secure fashion the cryptographic credentials it receives from the ID consolidation service and from multiple ID providers. Keys stored in the CCS should not be exported even in the event that the OS get compromised. This can be achieved by using a Secure OS (also referred as Trusted OS) along with hardware to set a Trusted Execution Environment (TEE).

TEE in an emerging technology that comes built-in the newest mobile devices. It provides an execution environment with elevated security mechanisms. However, a major issue with the current state of TEE is the close and limited interaction with this technology. In particular, despite the widespread deployment of hardware-based TEEs in mobile devices, application developers have lacked the programming interfaces to use TEE functionality. Therefore, to the reasons described above, ReCRED will utilize Open-TEE.

Open-TEE is a virtual TEE that conforms to GlobalPlatforms specifications. The main goal of Open-TEE is to help programmers in the development of applications that will be executed inside TEE. Afterwards, the developed code can be compiled with the appropriate platform and be ported on an actual TEE environment. This means that the ReCRED TEE code will be easily ported to any commercial and non-commercial TEE.

Therefore, ReCRED will follow two different approaches for TEE:

1. We will take advantage of TEE functionality as deployed in Android-based mobile devices to develop and provide two important security mechanisms: a) Secure Credential Storage for the various components in the User device (e.g., face signatures, Idemix credentials, etc.) and b) Secure Key Management for FIDO protocol.
2. We will use an open-source project named Open-TEE [2] for the development of critical, from a security point of view, functionality for anonymous credential protocols (i.e., Idemix and U-Prove) that will be executed inside TEE.

The first approach has been described in the first version of this deliverable (i.e., D2.3). Regarding the second approach, the Open-TEE will be used to develop a trusted application that will execute the cryptographic functions of Idemix and U-Prove at the side of the mobile device. A major outcome of this effort will be that the developed application will adhere to the specifications of GlobalPlatform for TEE applications. The key characteristic of Open-TEE is that it is not intended to emulate any specific TEE hardware. Open-TEE meets its goal of guaranteeing that trusted applications developed using it will compile and run on any GP-compliant TEE hardware. This makes the development of applications easier, since we have access to Software Development Kit available in Linux OS. As mentioned in [2], with Open-TEE, the complexity of the system is hidden from its users. They are presented with an SDK that exposes the Client and Core APIs without being required to have a deep understanding of how the overall framework works, thereby allowing them to focus on the development of their own trusted applications.

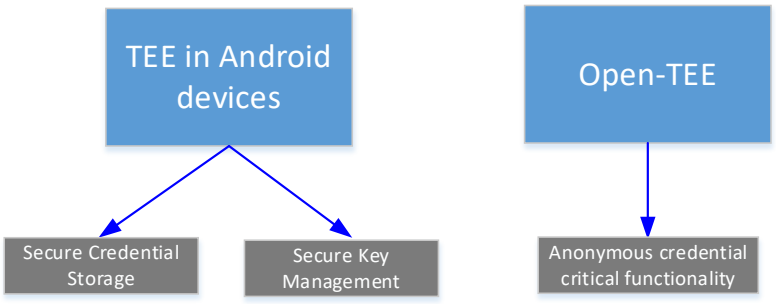


Figure 4: Two approaches for TEE functionality in ReCRED

2.2 Identity consolidator

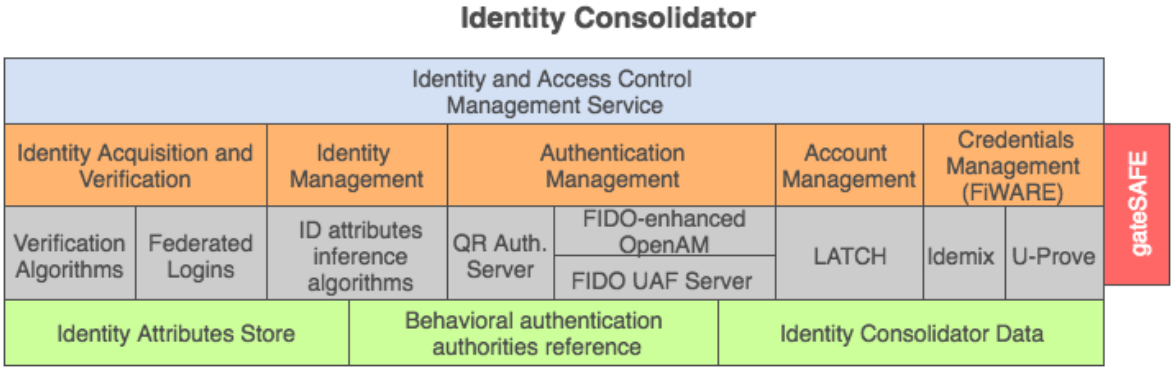


Figure 4 Identity consolidator protocol stack view

The Identity Consolidator is an integral component of the ReCRED ecosystem. It is responsible for horizontally binding the online identities of a user and vertically binding the real-world identities of a user to verifiable identity attributes. The ID consolidator exchanges verified identity attributes with the ID providers using **federated logins protocols** (such as OpenID Connect [25], OAuth [27][28]).

The consolidator offers an identity management service which allows users to voluntarily submit identity attributes and proofs of account ownership. The user interacts with this service using his identity management application. This service contains the **identity acquisition and verification** components which are responsible for acquiring and verifying all the personal identity information of a user.

The identity acquisition module uses smart trusted-computing-enabled devices (e.g., mobile devices) which will acquire the physical characteristics and the identity documentation of the users. The devices also use trusted software paths in order to securely and verifiable capture through the device’s camera images of a user and his identity documents. Additionally, the physical characteristics of a user (such as location) will be extracted exploiting the existing technology on the devices. All this information is then transferred into the identity verification module. This module contains **verification algorithms** that aim to verify the collected identity information of the user. These processes include automated verification (such as OCR) or peer-to-peer (crowdsourced) verification. Automated verification (such as face recognition and optical character recognition) is established on the acquired photos of the user. Peer-to-peer verification (such as crowdsourcing techniques) also takes place to verify that the information on the acquired photos match the declared personal identity information and physical characteristics. As soon as the verification is completed, all the verified independent identity attributes are stored to the identity attributes store module.

The consolidator can also infer identity attributes by aggregating the ID information of a user.

The ID consolidation service also enables ABAC with **the Cryptographic Credentials Management module**. This module allows users to issue cryptographic credentials from their verified identity

attributes, depending on their access needs, directly to their mobile device and then use them to access a variety of Service Providers. The credentials can also be backed-up with the consolidator, if the user desires to do so, for the purpose of failure recovery. Backing up the credentials is also useful because it allows the consolidator to prove that a user is who he claims to be while the user is offline. The cryptographic credentials issuance will be realized by using the **FiWARE** [31] interface that allows the seamless use of the **Idemix** and **U-Prove** cryptographic stacks.

In addition, the Identity Consolidator runs secure protocols directly with the Identity Providers through a well-defined federated login API to associate a user’s device with the identity attribute for which it issues a cryptographic credential. For more details see the ABAC diagram (Figure 14).

Considering the device-centric authentication, the ID consolidator acts as a mediator between Service Providers (relying parties) and the user. When an online service asks for a second-factor authentication, the user’s device redirects the service to the ID consolidator which informs the Service Provider, about which Behavioral Authentication Authority should be contacted for each second-factor authentication. This is realized through the **Behavioral Authentication Authorities Reference** store.

Furthermore, the Identity Consolidator offers various diverse authentication mechanisms for the authentication process with the consolidator. Specifically, the **Authentication Management Module** encapsulates a **FIDO-enhanced OpenAM** (acts as the OpenID Connect Provider) for undertaking FIDO Authentication. The authentication module offers various ways of authentication such as FIDO UAF, Mobile Connect + FIDO and master password. The authentication mechanisms are described in more detail in a dedicated subsection below.

The ID consolidator encapsulates the **Account Management Module** where **LATCH** can be activated [20] (see section 3.2.9) upon notifications that a behavioral authentication has failed, which may indicate that the device is no longer held by its legitimate user. In this case, LATCH locks the user out of all his logged in services. LATCH can also be activated by the user manually though the ID consolidator.

Last, the ID consolidator includes an **Identity Management Module** where the user can manage various aspects of his identity. More specifically, the user is able to: create verifiable profiles, set fine-grained consent management policies, view risks indicators for de-anonymazition, etc. More information about the Identity Management Module can be found in a dedicated section below.

2.2.1 Physical Identity Acquisition Module

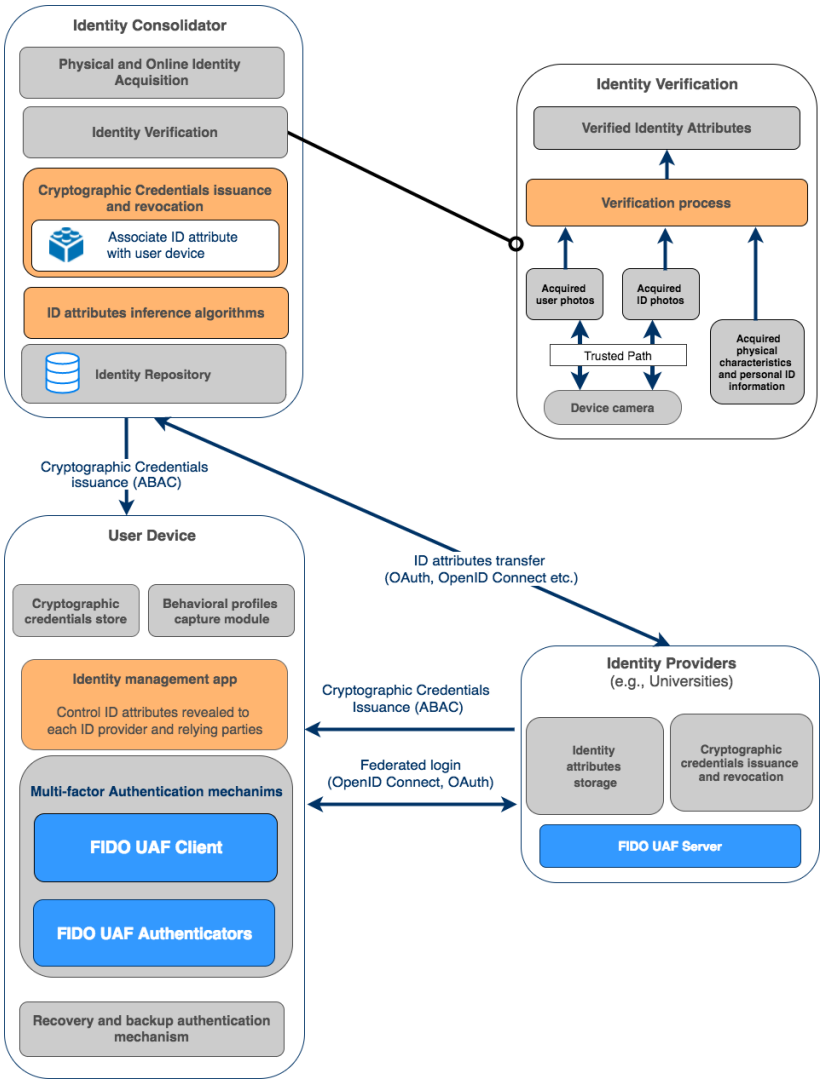


Figure 5: Identity consolidation with identity acquisition and verification component view

The Physical Identity Acquisition Module performs vertical identity binding. It is responsible for binding the real-world identity of a user to verifiable identity attributes. This module will be implemented as an application on smart trusted-computing-enabled devices and as a web application.

The Physical Identity Acquisition Module consists of the Identity Acquisition Process and the Identity Verification Process. The Identity Acquisition Process involves the acquisition of physical characteristics of the users as well as physical identity documentation. This process uses trusted paths on the devices in order to securely capture images of a user and his physical characteristics (e.g., location). The identity acquisition module uses smart trusted-computing-enabled devices (e.g.,

mobile device) which will acquire the physical characteristics and identity documentation of the users. The devices also use trusted software paths in order to securely and verifiably capture through the device’s camera images of a user and his documentation. Additionally, the physical characteristics of a user (such as location) will be extracted. All this information is then transferred into the identity verification process.

The Identity Verification Process uses crowdsourcing techniques and automated means in order to securely verify the acquired images and physical characteristics, while preserving the privacy of the users. Figure 6 shows how the physical identity acquisition process works and how a user declares and verifies attributes of his real-world documentation. The two processes of the Identity Acquisition Module are explained, in more detail, in subsequent sections.

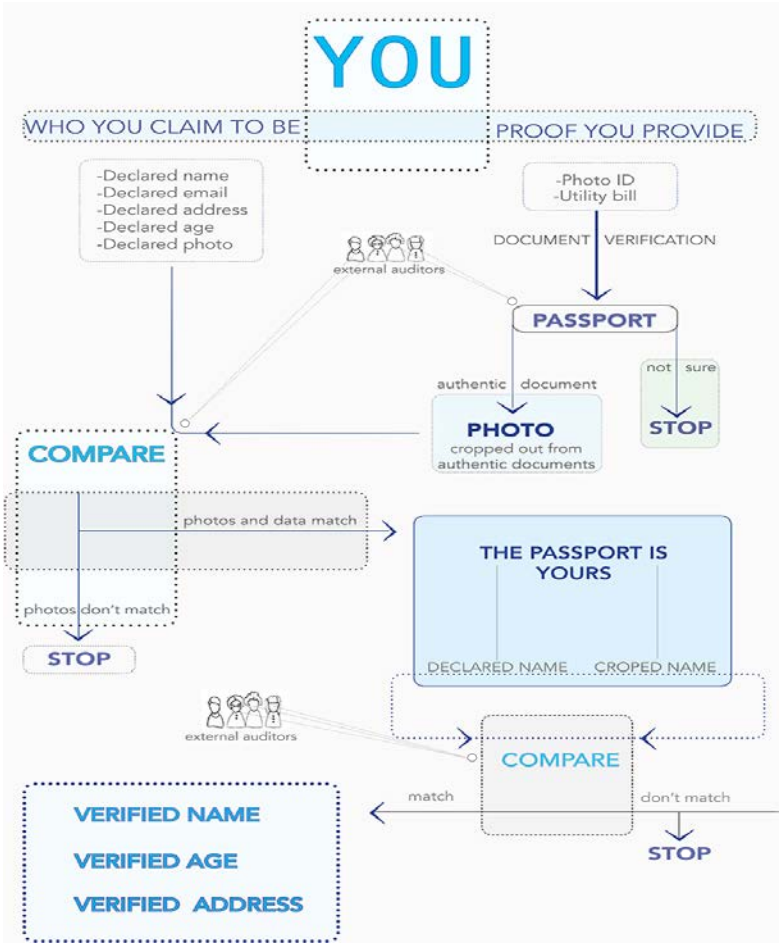


Figure 6 Physical Identity Acquisition and verification process

2.2.1.1 Identity Acquisition Process

Initially, the Identity acquisition process requires from the user to declare their identity information like their name, surname, identity number, date of birth, etc. through a Web or mobile interface. The acquisition process also involves the acquirement of physical identity documentation of the user. The user has to capture, through his device’s camera, images of their real-world identity and their face. Afterwards, the process enables the extraction and verification of the user’s desired identity attributes (such as identity number, full name, date of birth). This is achieved, by requiring from the user to crop the corresponding parts from the captured image of his identity. Furthermore, this module enables the acquirement of additional physical characteristics of the users (e.g., his address). Some of those characteristics may require additional verification that will be described in the identity verification process.

The identity acquisition process periodically captures behavioral characteristics of the user (e.g., location) for verification purposes that will be explained further below.

Furthermore, the identity acquisition will integrate the IRMA findings in order to acquire identity attributes from NFC-enabled cards. This will enable the secure and easy acquirement of identity attributes.

2.2.1.2 Identity Verification Process

The Identity Verification process receives all the acquired identity information of the user from the identity acquisition process. This process uses peer-to-peer verification and automated means in order to securely verify identity information. Specifically, the process exploits the following well-known techniques:

- Face Detection and Recognition: this method is used for the verification of the captured images of the user. At first, face detection is used to verify whether a face is included in the acquired user photos, both documentation and face photos. As soon as a face is detected in the images, then we perform face recognition. Face recognition is used to check whether a user photo matches to his securely captured identity photo.
- Optical Character Recognition (OCR): this is performed on the acquired user photos. Specifically, it is performed to the cropped photos in order to first verify that they contain a text and then extract it from each photo. In the end, those texts are compared with the user’s declared identity information.
- Peer-to-peer verification: crowdsourcing techniques are used to verify that the information on the acquired photos match the declared personal identity information and physical characteristics. The verification takes place via crowdsourcing it to other users of the platform. Confidentiality is preserved using watermarking and cropping techniques to ensure that nobody has access to reusable or forgeable photos of the users’ physical documentations.

As soon as the verification has been completed, all the verified identity information is transformed into independent identity attributes that are stored into the Identity Repository of the Identity

Consolidator platform. More details about the Physical Identity Acquisition Module can be found in D4.1.

2.2.2 Online Identity Acquisition Module

The Online Identity Acquisition module obtains identity information from various ID Providers, such as online social networks like Facebook and Google+, using Facebook Connect and OpenID Connect/OAuth2. The main challenge in developing this module is to acquire user information integrated from a user’s online accounts that user wishes to connect to the IDC. Once the IDC obtains access, it can collect the identity from the various ID Providers and it processes this information using the Identity Integration Module to determine their validity.

2.2.2.1 Online Acquisition process

This module is responsible for horizontally binding of all online user accounts. For each Identity Provider an online authentication process is required. The user should also give explicit authorization to each service to access the user’s personal information. Following the authentication process, the attributes acquisition takes place. Attributes such as date of birth, location, education, occupation, etc. will be retrieved and will be stored to the Identity Repository of the IDC. Figure 7 shows how the Online Identity Acquisition module retrieves identity information from the social accounts of the users.

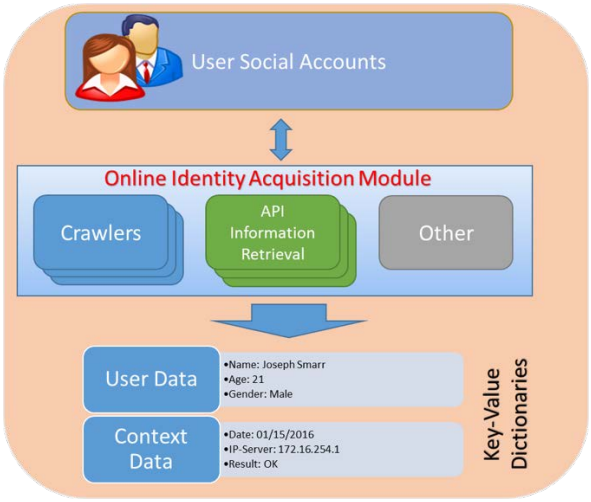


Figure 7 Online Identity Acquisition Module

2.2.2.2 Identity Integration Module

This module is responsible for standardizing and normalizing user information. It manages the spectrum of information elements that a user has on social networks, blobs, portals or systems. A typical user has to deal with multiple digital identities which usually are stored in social networks

and managed independently of one another. Identity Integration is the process of providing a unified view of the data spread across different sources.

To integrate user profiles from different social network, the usual process consists of three major phases: data extraction, data transformation and storage.

The data extraction will be provided through Identity Provider API or by the non-user assisted module. In all cases, after the extraction of the data the data will be transformed and stored on the database by calling the Storage API.

The identity integration module is responsible for aggregating and connecting the acquired online and physical user attributes, as well as for inferring the veracity of the claimed identity attributes via means of statistical data analysis techniques. The identity integration module can also infer missing ID attributes. The identity integration module is also responsible for assigning confidence scores for the veracity of the attributes and for labelling identity attributes based on their origin. For example, the Identity Consolidator should be able to tell verifiers that user A is more than 18 years old with confidence 90%. In order to achieve this, we make the following considerations:

1. We assume that each Identity Provider has a list of pre-defined attributes and a predefined Level of Assurance (LoA).
2. For each attribute that needs to be transferred to the IDC we make the following assumptions:
    a. If an attribute is present only in one Identity Provider, it passes immediately to the unified profile with a confidence score that matches the LoA of the Identity Provider.
    b. If an attribute is present in multiple Identity Providers that have different LoAs:
        i. If the attribute value is the same for all Identity Providers, the attribute is passed to the unified profile and the confidence score is equal to the highest LoA of the Identity Providers.
        ii. If the attribute value is not the same for all Identity Providers, we introduce the term Penalization Factor. In this case, the attribute with the highest LoA is selected and added to the unified profile and the confidence score is calculated by subtracting the highest LoA with the penalization factor. The penalisation factor (PF) increases with the number of mismatches in the attribute value, but will be in any case higher to the Highest LoA -1. For example, we have 3 Identity Providers (IdP1 with LoA = 4, IdP2 with LoA = 3 and IdP3 with LoA = 2). The consolidated attribute will be the one provided by IdP1. If IdP1 mismatches with only IdP2 or IdP3 the CS will be 4 - PF1 (e.g., PF1 = 0.25 and CS = 3.75). If IdP1 mismatches with both IdP2 and IdP3 the CS will be 4 - PF2 (e.g., PF2 = 0.5 and CS = 3.5).
    c. If an attribute is present in 2 or more Identity Providers belonging to the same LoA and there are discrepancies with regard to the value, there will be a predefined ranking between the Identity Providers belonging to the same LoA and the selected value of the attribute will be the one belonging to the highest ranked provided. For mismatches we will use a penalisation factor similarly as explained in the previous case.

More details about the Online Identity Acquisition Module and the Identity Integration Module can be found in D4.1.

2.2.3 Account Management Module

The Account Management Module within the ID Consolidator is responsible to manage the status of online accounts and to expose this information to authorized parties within the ReCRED framework.

The Account Management Module enables users to register a BAA or ID provider with the ID consolidator, and BAAs and ID provider admins to register their entities with the ID consolidator. It is also responsible to act as a BAA discovery service for Service Providers that require second factor device-to-service authentication.

Furthermore, the Account Management Module is considered as the main recovery mechanism for restoring users’ accounts. Specifically, the Account Management Module stores information regarding the URL of a registered IDP and the respective username. The Account Management Module initiates the recovery procedure as described below:

- If the Credentials Management Module doesn’t have the secret key for a respective IDP then it initiates the IDP-specific account recovery procedure.
- If the Credentials Management Module has the secret key for a respective IDP then the Account Management Module retrieves the secret key and it stores it to the new device. In this case, the IDP is agnostic of the recovery procedure.

The functionalities supported to the end-users that are related to the Account Management module.

- Identity Provider Registration: The user will be able to easily create an account to the ReCRED’s ID consolidation service for the Identity Provider.
- Deletion of user's account from the ID Consolidator: The user will be able to delete his account from the ReCRED’s ID consolidation service
- Recover accounts: The user is presented with a list of the accounts that he have and from there he is able to recover the secret keys (e.g., private keys) to a new device (e.g., after a device failure)
- Register Behavioral Authentication authority: The user can inform the ID consolidator which BAA authenticates him.
- Managing behavioral profiles: A user should be able to fully control the behavioral profiles they will provide to the ReCRED platform. They will be able to configure their settings depending on their needs, through this configuration interface. Note that the content of the behavioral profiles will not be stored on the IDC. Instead, the Account Management Module will redirect the user to the BAA where he will perform view and manage operations.
- Latch/ Unlatch online accounts: The user is presented with a list of all the connected online accounts and the status of each account (latched / unlatched – those terms could be replaced by locked / unlocked). The user can also manually latch an unlatched account or

unlatch a latched one. Furthermore, he should be able to set policies to automatically lock and unlock accounts.

- View history of Latch changes: The user can see the history of all the latched / unlatched services. It includes both changes applied manually by the user or automatically by the IDC.
- Continuous Authentication using BAAs and Latch: In addition to on-demand behavioral authentication where the BAA acts as an ID provider, the BAA also performs continuous authentication (i.e., using the browsing behavior), while having an open session with the IDC (account management module), and proactively Latches out the accounts when needed.
- Define Account Locking Policies: A user may define arbitrary policies to automatically lock or unlock an account. For example, a user may define a policy to lock his corporate email account over weekends and to lock his e-banking account at night. ReCRED also allows the ID Consolidator to act on behalf of the user and lock his online accounts if the ID Consolidator detects a high risk of account compromise.
- Configure Degree of Privacy: Users should be able to configure their account with the Identity Consolidator. An example is when a user wants to increase/decrease the levels of privacy within the IDC service. A user will have the option to choose between highest and lowest degree of privacy. The different degrees of privacy are described below.

Highest degree of privacy: In this case, the consolidator acts as a discovery service under a Federated Identity model. Specifically, with the exception of the user’s real life name, surname and Identity document number, the consolidator can only store the information that a user has an attribute and the respective Identity provider that holds it. When a Service Provider needs an attribute, the consolidator redirects him to the respective Identity Provider. In such way, the Service Provider obtains the attributes without the need of revealing any attributes to the ID consolidation service thus providing a more privacy-preserving solution.

Least degree of privacy: In this case, the consolidator will store a user's attributes on his service as well as credentials to access external service providers. For this feature, the Account Management Module interfaces with the Identity Profile Management and the Credential Management module.

2.2.3.1 Identity Federation

The Account Management Module works together with the ID Profile Management module to provide Federated ID services by enabling ID providers to query the IDC for certain attributes that the Service Providers have requested, and they do not maintain themselves. The IDC either responds with the attribute value in case it stores it, or redirects the requesting Service Provider to the appropriate ID provider who maintains that attribute. In other words, this module acts as an attribute discovery service to instantiate a Federated Identity solution.

The Identity Federation is used when a Service Provider asks an Identity provider for a proof of identity, and the Identity Provider does not know that identity. In this case, the Identity provider redirects the user to the IDC for further actions. Subsequently, the IDC either responds with the identity attributes (if he has the attributes, and the user trusts the consolidator) or redirects the

Service Provider to the appropriate Identity Provider who maintains the requested attributes. The Identity Consolidator acts as an Identity provider discovery service, for the Identity Federation.

The Identity Consolidator has a complete list of attributes for each Identity Provider. In the case the Identity Consolidator does not know the value to the user's attribute, it will notify the primary Identity Provider to identify other Identity Providers the Service Provider should contact.

The Identity Federation will be the only way that the IDC interacts with the user and the IDPs in case of the highest degree of privacy. However, the Identity Federation will also be used in less than the highest degree of privacy modes, as it will not always be the case that the user has transferred all his attributes from the IDPs to the IDC.

2.2.3.2 LATCH

The basic functionality of LATCH [20] is to temporarily enable/disable multiple third-party accounts belonging to a user. Other functionalities of LATCH (e.g., one-time passwords) are out of scope. LATCH does not replace identity management at Service Providers and Identity Providers. That is, a third-party server must still implement and manage its own mechanisms to authenticate users. Nevertheless, LATCH provides an additional security layer atop of the authentication mechanism of the third-party server. In order to use LATCH, a user must create an account with the LATCH server. A user can “pair” his third-party accounts to his LATCH account. The third-party server must support LATCH. Third-party accounts within the LATCH account of a user can be enabled/disabled (or latched/unlatched) at will. When a third-party account is latched by a user, the third-party server does not accept login attempts on behalf of that user. In order for the server to accept login attempts, the account must be un-latched.

A third-party server must register its application with LATCH. The server receives an applicationId that identifies its application within the LATCH ecosystem. The server also receives an applicationSecret that is used to authenticate requests made to the LATCH server. Similarly, users must create a LATCH account and receive a unique userId and a userSecret that is used to authenticate requests made to the LATCH server. For both scenarios, the algorithm used to authenticate requests is HMAC-SHA1.

In the following we list the most common operations within LATCH.

2.2.3.2.1 Account Pairing.

Pairing of a third-party account with a LATCH account requires the server to receive a token from the user. The token is a 6-char random string that the user can obtain from the LATCH application on his device. The third-party server exchanges the token for a 64-char accountId that uniquely identifies the LATCH account that is being paired with the third-party account.

2.2.3.2.2 Check Status.

Third-party servers that have paired a third-party account to a LATCH account, can query the LATCH server for the status (e.g., latched or unlatched) of the third-party account. The query must include the accountId obtained during the Account Pairing procedure.

2.2.3.2.3 Change Status.

A user can latch/unlatch each of his third-party accounts. Some applications may also enable latching/unlatching of single operations. For example, an online banking application may allow user to latch/unlatch international wire transfers.

2.2.3.2.4 Latch Support Tool.

In the context of ReCRED, we decided to re-create a subset of the features provided by Latch, tailor them to the ReCRED needs, and add functionalities required to fulfil the project goals. Our own version of LATCH is referred to Account Locking Mechanism. The main rational behind our choice is that the current version of LATCH does not allow one party to latch/unlatch all the accounts of a given user. Whereas our goal is to enable the ID Consolidator to latch all accounts of a user, upon detection of suspicious activity related to any account of that user. Our Account Locking Mechanism mimics the functionalities of LATCH as described above, but it further grants to the ID Consolidator to manage all the account of a user.

2.2.3.3 Mobile Connect Proxy Service

For this module, the Service Provider will only trust the Identity Consolidator as the contact point to the Mobile Connect Telco providers federation that determines the Telco Provider for a given identifier. Additionally, the Identity Consolidator will act as an OpenID Connect ID Provider that stores attributes retrieved via Mobile Connect sessions.

The Identity Consolidator invokes the GSMA apigee API Exchange-enabled discovery service on a trusted Mobile Connect Telco provider. The API Exchange is used as the federation mechanism for Mobile Connect. It ensures that Service Providers (in this case the IDC) only need to connect to one operator of a federation to get access to customers of all connected operators. Therefore, the Identiy Consolidator as a Service Provider, uses the GSMA apigee API Exchange Discovery Service of the trusted Mobile Connect provider to ask for the Telco Provider for a given phone number or IP or MSISDN, etc. The Identity Consolidator will call the apigee API of the discovered Telco provider to initiate a Mobile Connect session between the user and the Telco Provider.

The next steps will be for the user to authorize the transfer of attributes between the Mobile Connect Provider and the Identity Consolidator, and the IDC will store those Mobiles Connect-retrieved attributes. Finally, the IDC then continues to act as the ID Provider for the OpenID Connect session with the non-Mobile Connect-enabled Service Provider.

2.2.4 Credential Management Module

The Credential Management (CM) module is responsible for the issuance of cryptographic credentials to the devices of the users. The main challenges on the design and the development of this module are: i) to securely map identity-attributes, acquired from consolidated identities, to cryptographic credentials to be issued to the user and ii) to provide interfaces and functionalities to make it possible to issue credentials from different sources (i.e. heterogeneous third-party Identity Providers). The CM module will initially integrate Idemix and U-prove cryptographic engines for credentials issuing and will be an extension of the functionalities provided by the ReCRED credential issuance module running on the ReCRED Identity Providers. The credentials include a set of cryptographic attributes which can be either:

1. Stored on the ReCRED Identity Consolidator, i.e. extracted from already verified identity-attributes coming from a previous physical identity acquisition or online identity consolidation.
2. Managed by a third-party Identity Provider (e.g. University, Bank, Municipality, etc.) which:
    - **Does not support** the issuance of cryptographic credentials and that rely on the Identity Consolidator credentials issuance capabilities.
    - **Supports** the issuance of cryptographic credentials to which the user device is redirected (by the Identity Consolidator) when requesting credentials.

In order to allow the issuing of consistent credentials and the related definition of policies at the Service Providers, the Identity Consolidator should allow to define (or acquire) a set of supported credential templates that define how the credentials are compiled (i.e. attributes required to fill-in the credentials) and how to treat them for policies definition. When the user interacts with the Identity Consolidator, she can select one of these formats and which of her acquired and verified identity-attributes are suitable to be included in the issued credential.

We assume that the user is already authenticated with their personal device which is in turn authenticated with the ReCRED Identity Consolidator. As an alternative, the user could authenticate and make use of the web interface to trigger credential issuance to its own personal device. The user should trust the Identity Consolidator and the appropriate identity-attribute matching checks are performed before the actual credential issuance process is started.

To design a complete solution able to fulfill such requirement, the Credential Management module should implement functionalities that allow to:

1. Extend the Identity Consolidator functionalities in order to map acquired identity-attributes to cryptographic credentials.

2. Provide a trusted centralized component in the ReCRED architecture that provides support to the issuance procedure of third-party Identity Providers.

In any case the credentials are finally stored on the user device and are optionally (and based on user selection) backed-up into the Identity Consolidator. Indeed, the Credential Management

module should support the secure backup (to restore in case of lost user device) of the user’s owned credentials, by providing strong encryption that uses keys known only to/obtainable only by the user. In Figure 8 the high-level architecture of the Credential Manager module and the related interfaces to the involved components of the ReCRED framework.

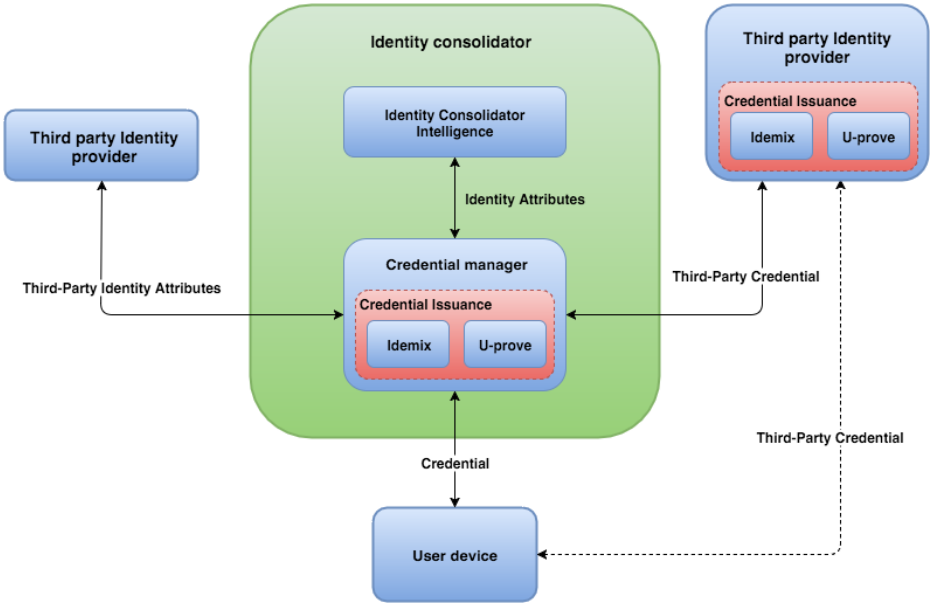


Figure 8 Architecture and Interfaces for the Credential Issuance service provided by the Credential Management Module

The Credential Management module should extend the Identity Consolidator functionalities providing support for credential issuing from heterogeneous third party Identity Providers in different ways:

- The Credential Management module may serve as a trusted Identity Portal for External Identity Providers supporting ReCRED credentials issuance. Indeed, both the user requesting the issuance of a credential and the involved Identity Provider can exploit this functionality to request the Identity Consolidator to authenticate related parties. The Identity Provider and the Identity Consolidator exchange user authentication/sign-on information. Then the user is redirected to the Identity Provider, which does not require a new sign-on or identity verification and is thus able to issue directly (or by means of the Identity Consolidator) credentials to the user.
- The Credential Management module may provide to the user the functionality of trusted redirect to Identity Providers supporting ReCRED credentials issuance. The user contacts the Identity Consolidator in order to request a credential from an Identity Provider, as above. In this case the ID Consolidator does not directly interact with the Identity Provider but it simply redirects the user to the Identity Provider. Since there is no direct interaction between the Identity Provider and the Identity Consolidator, the user has first to authenticate to the Identity Provider. The issuance procedure then occurs directly between the Identity Provider running the ReCRED issuance module and the user.

- Identity Providers that do not support the ReCRED issuance module can take advantage of the Identity Consolidator proxy feature to issue credentials through the Credential Management module. In this case, the user contacts the Identity Consolidator and requests the issuance of a credential from an Identity Provider which does not support the ReCRED credential issuance module. The Identity Consolidator interacts with the external Identity Provider, acquires from it the appropriate attributes, and issues the requested credential through the Credential Management module on behalf of the third-party Identity Provider.

The supported operations to the end-users are the following:

- Issue cryptographic credentials: The user is presented with a list of supported credential templates and he can issue a new cryptographic credential based on the selected template. Cryptographic credentials can be issued either directly by the IDC (as long as the user trusts the IDC) or directly by a selected IDP which can issue a credential for the selected template. Alternatively, a new cryptographic credential can be issued automatically, whenever required, in order to grant access to an online service, according to specific rules defined by the provider (e.g. a proof of age).
- List supported attributes per IDP: The user is presented with a list of all the IDPs and for each IDP a list of all the supported attributes.
- List issued cryptographic credentials: The user is presented with a list of all the cryptographic credentials that have been issued to their device.
- View issued cryptographic credentials’ details: The user can select an issued cryptographic credential, in order to see extended details for the selected credential (issued date, expiration date, issuer, whether it has been backed up to the IDC, a history of the credential’s uses, etc.).
- Reissue expired cryptographic credentials: The user can select an expired cryptographic credential, in order to reissue it. A new credential is issued, by the same authority and with a new expiration date. Alternatively, the user can activate an option (through the user settings) authorizing the IDC to automatically attempt to reissue expired certificates.
- Encrypt/Decrypt cryptographic credentials: The user can select one or more credentials, in order to encrypt them or decrypt them.
- Backup cryptographic credentials: The user can select one or more credentials that have not been backed up to the IDC, in order to back them up. All the selected credentials are transferred to the IDC through a secure channel. Alternatively, the user can activate an option (through the user settings) to automatically backup issued credentials to the IDC.
- Restore cryptographic credentials: The user can select to restore any cryptographic credentials that are backed up to the IDC and not stored in the user’s device (useful when a device is reset or a new device is purchased).
- Erase cryptographic credentials: The user can select to erase some or all of the cryptographic credentials that have been issued. The user has the options to erase the credentials from the IDC, from her device or both.

The supported operations to the Identity providers are the following:

- Manage supported identity attributes: The IDP Administrator is presented with all the identity attributes used by the various credential templates and they can select which of those attributes are supported by the IDP. An IDP can issue cryptographic credentials only for templates for which all the required attributes are supported.
- Issue cryptographic credentials: The IDP Administrator can manually issue a credential and send it to the end user’s device (e.g. in the case of a university administrator who registers students and professors to the campus Wi-Fi). A manually issued credential may or may not have an expiration date.
- List issued cryptographic credentials: The IDP Administrator is presented with a list of all the cryptographic credentials that have been issued.
- Revoke cryptographic credentials: The IDP Administrator can select a specific issued credential, in order to revoke it. The credential is erased from the user’s device and/or the IDC (or it is marked as “revoked”).
- View statistics: The IDP Administrator can see statistics regarding cryptographic credentials issued by the IDP.

2.2.5 Authentication Management Module

The ID Consolidator offers a versatile Authentication Management Module (AMM), which supports multiple authentication methods, with each method offering a different Level of Assurance (LOA) and allowing access to different categories of the user’s data and different actions. The supported authentication methods comply with the following Levels of Assurance defined by the National Institute of Standards and Technology (NIST), in terms of the likely consequences of an authentication error. As the consequences of an authentication error become more serious, the required level of assurance increases. Of course, the token methods allowed for each given level are also allowed for all of its lower levels.

Using the Authentication Management Module mobile application, the user has to register with the Identity Consolidator (IDC), up to a given authentication LoA which depends on the proofs of his identity he has provided and whether his device and Identity Providers support the appropriate soft or hard cryptographic tokens. This can be achieved by using the Physical Identity Acquisition Module to provide proofs of real-life identity, combined with proofs from the Online Identity Acquisition Module. Upon registration, the Identity Consolidator issues' FIDO credentials on his device.

Alternatively, a user may register at a given LoA if he provides via OpenID Connect proof that he owns an account with a certain Identity Provider (IdP) for the given LoA and the IDC trusts that IdP at this level. For example, the IDC may trust a Telco to provide LoA 4, so a user will be considered LoA 4 if he uses Mobile Connect to connect his Telco account with the IDC and uses OAuth2 to transfer his real-life identity information from the Telco to the IDC.

The user authenticates to the IDC primarily by using his IDC-issued FIDO credential. Depending on the authentication method, the user is authenticated up to a certain LOA, which is decided by the

IDP following the NIST guidelines[1]. For example, if the user authenticates to the IDC solely by using his IDC-issued FIDO credential, he is considered authenticated at LoA 3. If the authentication of the user is undertaken using the FIDO credentials that are stored in the TEE capabilities of the user’s device and are protected by the hardware, then the authentication is considered LoA 4. Note that the FIDO UAF specification is considered as a multi-factor authentication mechanism as it requires a local authentication using biometrics or pin to unlock the keys and public key cryptography for remote service authentication (something you are is the biometric and something you have are the FIDO keys).

The LoA can be achieved as follows:

- Password tokens can satisfy the LoA 1 and LoA 2.
- Soft cryptographic tokens may be used for LoA 1 to 2,
- If soft cryptographic tokens are used for multi-factor authentication then LoA 3 can be achieved (e.g., FIDO authentication).
- If hard tokens are used for multi-factor authentication then LoA 4 can be achieved (e.g., FIDO authentication with keys stored in TEE).

The chosen IdPs and the method via which authenticates to it determines the LoA. As soon as the user is authenticated and the LOA has been determined, then he can view, transfer or delete attributes depending on his session’s LoA. In particular, the Identity provider may optionally set policies to restrict the access to attributes by the user, depending on his session LoA.

Please note that as soon as the user is authenticated to the IDC and a certain IDP, the user, through the Identity Profile Management Module, can initiate the transfer of identity attributes from the IDP to the ID Consolidator. Importantly, those attributes are labeled with the minimum LoA of the LoA they had at the origin IDP and the LoA of the user session.

The AMM also provides some recovery mechanisms for cases where users are unable to login to theirs ID Consolidator accounts. Such mechanisms consist of a set of security questions that were defined during the registration with the ID Consolidator, SMS verification and prove of possession of online accounts (such as Facebook, Google, etc.,) via well-known protocols (such as OpenID Connect, etc.).

The AMM implementation will be based upon the OpenAM identity solution. All the aforementioned features will be checked against the current implementation of OpenAM and we will provide extensions to cover all the scenarios that are required by ReCRED.

2.2.6 Identity Management Module

The Identity Management module is a common framework that serves as a standard for the definition and representation of user identity attributes within a given online service. Identity Management module is subdivided in two sub-modules, the identity management and the consent

[1] Burr, William E., et al. "Electronic Authentication Guideline: Recommendations of the National Institute of Standards and Technology-Special Publication 800-63-1." (2012).

management. In general, it provides an identity matrix containing the different type and range of identifiers, and unique identity attributes a user can have. A representative use case is **reputation** in a certain online platform such as eBay, which we view as an attribute of that user. This identity attribute is currently used only by eBay and its users. Apart from that, this module offers a protocol to transfer identity attributes between ID providers and at the same time guarantees the security in the transfer of such sensitive information. Also, it gives users the option to create partial verifiable profiles, which consist of selected identity attributes of a user, to be presented to verifiers depending on the context and the access control requirements. Furthermore, it allows users to define their consent for the management of their various identity attributes.

2.2.6.1.1 Identity Profile Management Module

This sub-module provides the user’s interface that allows the user to know and manage what each identity provider and Service Providers knows about them. It enables the user to transfer attributes between ID providers and between ID providers and the IDC using OpenID Connect sessions in which the destination ID provider acts as the Service Provider. When transferring attributes, the system abides to the policies defined within the identity consent management module. The transfer of attributes from ID providers to the IDC is performed by invoking functionalities of the online ID acquisition module. Furthermore, the user has the ability to delete an identity attribute from an IDP.

The user can only view, delete and insert new validated attributes that are accompanied with a certain LoA (see Authentication Management Module). The LoA of an attribute depends upon the identity proofing mechanism that was used to verify it. The LoA of a user session depends upon the authentication mechanism. The LoA of the transferred attributes is decided by the consolidator by taking into account the minimum LoA of the attribute and the user session. For example, a LoA 4 attribute transferred during a LoA 3 user session, is stored as a LoA 3 attribute at the destination ID provider.

It also enables the user to determine the risk of identity providers and Service Providers inferring information about them that they did not explicitly reveal to them. The value of identity attributes that were not explicitly revealed to an ID or Service provider can leak by statistically analyzing correlations between identity attributes, thus the risk will be calculated by using similar techniques.

Lastly, this module provides the required functionality to the user to create and manage partial verifiable profiles as described below.

2.2.6.1.2 Consent management module

The Consent Management module allows users and IDPs to define their consent for their various identity attributes. Furthermore, the consent management module is subdivided into the following:

- **User-defined policy for attribute transfer and proof:** Such functionality is used when the user wants to define policies about to which IDPs and Service Providers, their attributes should be revealed. The Consent Management module will provide the user with the ability to involve the LoA in the policy decision process. For example, the user may define that it

does not wish to reveal their address to online social network services or any service provider under certain LoA.

- **Identity Provider-defined policy for attribute transfer and proof:** It is responsible for obtaining policies (with respect to what identity attributes can each Service Provider see) for individual attributes from identity providers ensuring that the Identity Consolidator reveals attributes to Service Providers according to these policies. The Consent Management module will provide the identity provider with the ability to involve the LoA in the policy decision process. This is used, especially from ID Providers who do not wish certain attributes, or attributes with specific LoA, to be revealed to certain unauthorized parties or other identity providers. For example, the Social Security Administration (ID provider) provides the social security number that should be revealed only to reputable verifiers, such as banks. Or an IDP may decide that it does not want the attributes of its users to be proven using Idemix/U-Prove and that they should be proven through the IDP via OAuth instead (so that the IDP always knows where these credentials have been shown).

2.2.6.2 Identity Management

The Identity Management module contains a web and mobile application, through which the users can view and manage their own identity data, as well as the identity attributes that each identity provider maintains for them or any online service knows. Therefore, the application contains two main interfaces:

1. **Identity Data Management**: The user can view all current data that are stored in the Identity Repository. The user is also able to centrally update the value of some identity attributes, as long as this is allowed by the Identity Consolidator policies for these attributes. For example, the user cannot change their name or age (especially if these have already been verified) but they can change their current job or address details. An update on some attributes may or may not trigger the initiation of an identity acquisition and verification process (e.g. a new proof of address). After the data is updated (and verified, if necessary), all the online services that maintain the updated identity attributes are notified and the values they hold are automatically updated as well.
2. **Shared Identity Attributes**: The user can view all of the identity attributes that are shared with various online services. This can be achieved in two alternative ways:
   a) The user selects an identity attribute and all the online services that maintain that attribute are fetched and displayed.
   b) The user selects an online service and all the identity attributes maintained by that service are fetched and displayed.

2.2.6.3 Risk Management

The ID Management module will display to the user a risk figure indicating the possibility that an ID Provider or a Service Provider inferring the values of unknown user attributes based on the known user attributes that the ID Provider maintains for this user. The risk indicator is separate for each unknown attribute and ID Provider permutation.

Risk can be calculated only for attributes with known values i.e. attributes whose values are stored in the Identity Repository. For attribute values stored only in ID Providers it is impossible to determine their distribution and hence cannot calculate a risk factor.

A simplistic approach to risk calculation is to assume that risk is essentially the size of the population of user identities sharing similar attribute values over the entire population of user identities.

2.2.7 Storage API and Repository schema

The Storage API is the module that provides access to the Identity Repository to other components of the Identity Consolidator. Additionally, the Identity Repository is a central database where all the necessary information is stored. The Storage API interacts with the Identity Repository in order to perform CRUD operations on the repository. The Storage API will be exposed via a REST Interface and all the calls should be authenticated either by username/password or OAuth2.0 tokens.

The database schema should store at least the following:

1. User accounts information
2. Physical Identity information
3. Identity Providers information
4. User acquired locations information (for the purpose of verification)
5. Verification information for all the data
6. Audits information (for the purpose of verification)
7. Financial information (for the purpose of loan origination pilot)
8. Behavioral Authorities references
9. User to BAA and user to IdP mappings

Additionally, in this subsection, the conceptual model of the data associated to a profile account (contact or user) is defined. The design does not force a particular technology or a database schema model. The data modeling techniques are basically implementation agnostic. Our intention is that this model could be implemented using traditional relational database (Oracle, MySQL, etc.) or NoSQL database (HBase, MongoDB, etc.). It does not directly deal with issues of performance, scalability, cluster distribution and management, etc.

The ReCRED servers manage normalized user profile data in standard format. This makes it easier to parse and use the profile data without having to learn about each provider’s data format.

2.2.7.1 Database design

The Identity Consolidator deals with user identity information. This data comes from different sources and ID providers. For instance, users could have a different username in Facebook or LinkedIn. The database should be able to store and manage this diversity of data. It converts the data in one of the diverse data formats returned by the identity providers into a single standard

format. We call this process normalization. In addition, a user could provide direct information to ReCRED platform. This information, when validated, should be marked as original or preferred information.

The system will store user information using,

1. A user account table. This table will store information about the user account. This table will include a unique identification of the user (primary key) and other system related information like update datetime, if the user has been validated or the preferred user profile information (display name, gender, etc.)
2. Several tables with plural information. Plural fields are information elements with any number of instances per contact. Plural fields include emails, phone numbers, urls, addresses, photos, videos or media items, positions, etc. These tables store consolidated information and the source of this could come from identity providers, users accounts or introduced directly by the user.
3. Several tables with (social) accounts information. Users have external accounts like Facebook, Instagram or Twitter. These tables store consolidated information as provided by this accounts or external identity providers.

The following figure represents this model:

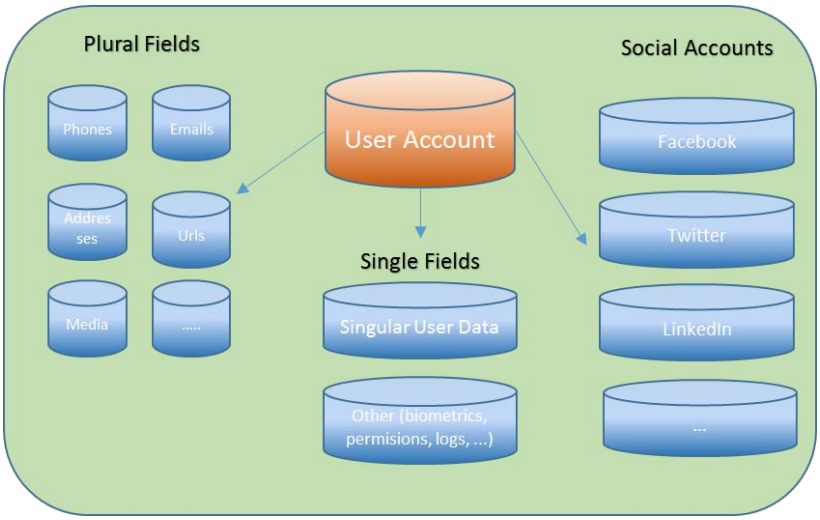


Figure 9 Storage Model

2.2.8 3rd Party API

The 3rd party API is used for communication purposes between the Identity Consolidator component of ReCRED architecture and other 3rd parties. Such parties include other ReCRED’s

components (e.g., Behavioral Authentication Authorities), Identity Providers and Service Providers that want to interact with the Identity Consolidator.

The 3rd party API offers, but is not limited to, the following operations:

- Service Providers should be able to interact with the Identity Consolidator component for the purpose of transferring trust between the services.
- Service Providers should be able to communicate with the Identity Consolidator component for the purpose of Two-Factor Authentication (2FA), if they desire. Specifically, the consolidator should be able to provide a reference to which Behavioral Authentication Authority (BAA) so that they can perform the additional authentication step.
- The BAA should be able to inform the Identity Consolidator component what behavioral aspects of each user they store. As a result, the consolidator always has a synchronized reference to the behavioral attributes that are stored in BAAs databases.
- The BAA should be able to inform the Identity Consolidator regarding the outcome (accept/reject) of an authentication attempt of a user to an online service. During this operation, the ID consolidator will check the defined securities policies and if are violated then it may lock some or all of user’s account.
- A Service Provider should be able to request the status of a user’s account. Specifically, a Service Provider should be able to query the Identity Consolidator which will reply with the account status (locked or unlocked).
- The ID consolidator should be able to issue cryptographic credentials directly to user’s devices. Additionally, if the user desire so, the consolidator should be able to back up those issued credentials.

2.3 Service Providers

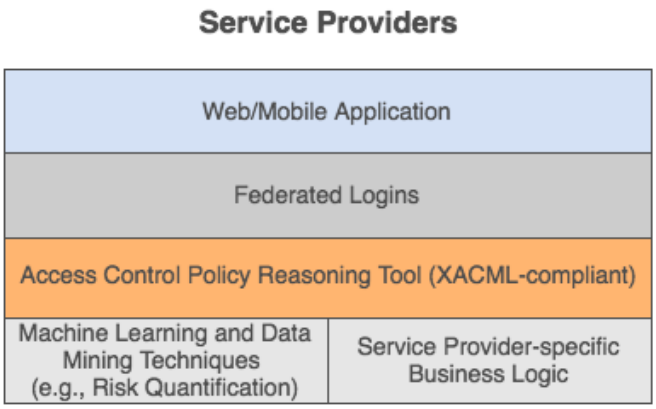


Figure 10 Service Providers protocol stack view

In the context of the ReCRED architecture, Service Providers are entities that are used by the users to gain access to various offered services. As can be seen by Figure 10, the Service Providers are simple entities that don’t encapsulate exotic protocols and interfaces (i.e., authentication mechanisms, cryptographic stacks, etc.,). This is an architectural decision that aims in maximizing the applicability of the ReCRED platform to Service Providers. In summary, the Service Providers will require none-to-minimal modifications in order to use and adapt to the ReCRED architecture.

The Service Providers, contain their **business logic** that defines their access control policy. For example, an online movie theater service may require from their users to prove their age before getting access to the service. This requirement is realized and offered by **the Access Control Policy Reasoning Tool** on the Service Providers. Specifically, this tool enables the definition of complex access control policies that enforce how access is granted. The Access Control Policy Reasoning tool on the Service Provider follow the XACML [32] specification, which is a well-defined specification for defining and evaluating complex access control policies and requests. Furthermore, the tool contains a recommendation engine, which make use of **Machine Learning and Data Mining techniques**, that aim to provide recommendations regarding the defined access control policies. Such recommendations may consist of: i) addition of new access control policies and ii) reduction of access control policies by deletion of overlapping or unnecessary policies.

In order for the Service Provider to identify a user, the authentication procedure is delegated to either the Identity Consolidator or to an Identity Provider using the OpenID Connect protocol (**Federated Logins**). Subsequently, the user authenticates to the Identity Provider which return the verified identity attributes. The user can also make use of cryptographic credentials so that it can prove the possession of a specific attribute. In this case, the user present it’s cryptographic credentials to the Identity Provider, which verifies them using the underlying cryptographic credentials stack (Idemix or U-Prove), and the verified attributes are delivered back to the Service Provider using the OpenID Connect specification.

The Service Provider can request for a second-factor authentication from the ReCRED platform. In this case, the Service Provider queries the Identity Consolidator in order to find an appropriate BAA that can provide a second-factor behavioral authentication. After receiving the information about the BAA, the Service Provider can query the BAA in order to verify if the user behave as usually, thus achieving second-factor behavioral authentication.

2.3.1 OpenID Connect Relying Party (Federated Logins)

The Service Providers would be able to delegate authentication and acquire identity attributes from an Identity Provider so the SP can grant access to the service to the users. This is achieved by having the OpenID Connect Relying Party code within its service so that he can query an Identity Provider for user attributes. Additionally, the Service Provider should maintain, for each Identity Provider, the following information:

1. A list of the Identity Provider’s OpenID Connect endpoints.

2. Client Id and Client Secret. These tokens are used by the Identity Provider so that it can identify the Relying Parties

In case that the Service Provider does not have this information for a particular Identity Provider, then it should perform a dynamic registration with the Identity Provider in order to acquire and store this information for future use.

2.3.2 Access Control Policy Reasoning Tool

Service Providers need to define the attribute-based policies for user access to services. The Access Control Policy Reasoning Tool aids them in the definition of these policies. The policies on the Service Providers should be compliant with the XACML standard (see Section 3.7). Furthermore, the access control policy reasoning tool offers a machine learning-based recommendation system that aims to provide recommendations for modifying the access control policies. More specifically, the reasoning tool offers the following:

- ABAC policies' creation: The Service Provider administrator will be able to define ABAC policies, in order to define how and when the users can get access.
- ABAC policies refinement request: The Service Provider administrator will be able to request policy refinement, so that the ABAC policies do not include overlaps.
- Denied requests search: The Service Provider administrator will be able to search the denied requests, so that a special permission on a specific user for a specific resource can be created.
- Special permissions' creation: The Service Provider administrator will be able to add permission to a user for a resource by creating a special permission.
- ABAC policy recommendation request: The Service Provider administrator will be able to request policy recommendation from the Machine Learning recommendation system, so that the ABAC policies of the system can be improved.
- Special permissions reduction request: The Service Provider administrator will be able to request special permissions' reduction, so that any special permissions that are already covered by the ABAC. policies to be removed.

2.3.2.1 Machine Learning Recommendation System

The Service Provider Access Control Policies Reasoning tool will be able to get recommendations about existing policies, and offer a better access control solution. The recommendation system should be an ML-based system (ABAC policy mining) where the system takes the active ABAC policies, the special permission's policies and the data from the authorization requests log files, as input and recommends new ABAC policies. The machine learning solution will consider the criticality of each resource, and with a general aim to minimize the overall number of special permissions that are active on the ABAC reasoning tool. The system should be able to expose these recommendations to the Service Provider Administrator front-end.

2.4 Behavioral Authentication Authorities

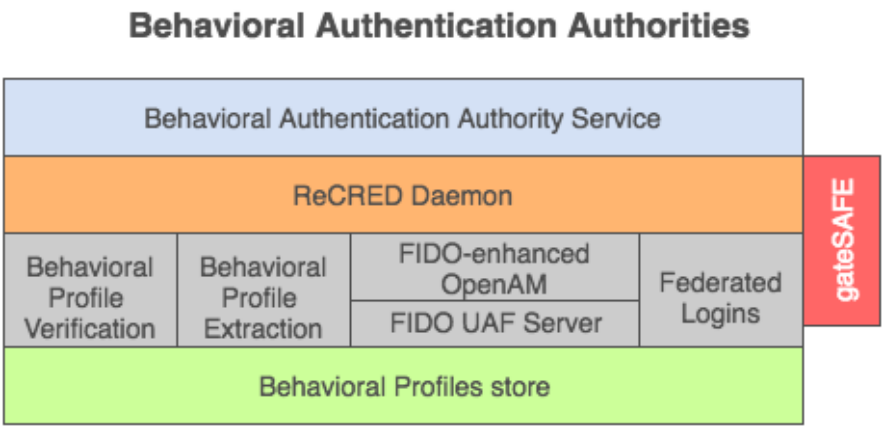


Figure 11 Behavioral authentication authorities protocol stack view

ReCRED aims to decouple the concept of behavioral authentication to a separate entity, called Behavioral Authentication Authorities (BAA), instead of encapsulating this functionality in the Identity Providers entity. This architectural decision has the following advantages:

1. Service Providers are able to explicitly request a specific 2nd factor behavioral authentication for a user. This can be achieved by querying the Identity Consolidator, which maintains the reference between the BAA’s, users and behavioral authentication method.
2. Identity Providers are not responsible for performing behavioral authentication, thus their architecture is simplified.
3. The BAAs are trusted entities, such as Telcos, which can easily track users’ behaviors such as the network activity. This empowers the whole ReCRED architecture as the behavioral profiles are securely stored on the databases of the BAAs.
4. By having a separate entity for Behavioral authentication, the ReCRED platform can offer, through the Identity Consolidator, exotic multifactor behavioral authentication. Furthermore, the ID Consolidator can enforce account locking policies based on the outcome of behavioral authentications.

The BAAs are essentially web-services that provide behavioral-based authentication and uses smartphones to harvest user behavior data in a way that is non-intrusive for the user and energy-efficient for the device. Behavioral Authentication Authorities instantiated by telecommunication providers can use network information to build the behavioral profile of a user. Incoming and outgoing calls, cell towers that “see” the device throughout the day, or web browsing habits, are some of the information that a BAA can use to authenticate an individual in a multi-factor authentication framework. Most importantly, such information can be collected by a telecommunication provider acting as a BAA in a manner that is completely transparent to the user. Call logs, cell tower connections and browsing histories are already being collected by telecommunication providers for security and billing purposes. ReCRED leverages such information

for authentication purposes. Other behavioral traits that ReCRED leverages require data collection directly on the smartphone, but are still transparent to the user. In particular, gait and typing behavior are used to verify the identity of the user. Both modalities collect data through the ReCRED app installed on the user device. While the user is walking or when she is typing, the ReCRED app collects data through the sensors available on the smartphone (e.g., accelerometer, gyroscope, etc.) and uploads them to the BAA. Uploaded data is used to build a user profile and, later on, to authenticate the user. Data upload only happens over a WiFi connection, not to affect the data allowance of the user. Another option would be for a telco-BAA to fetch such measurements in real-time, without charging the user for the consumed bandwidth.

The BAAs capture and store, on their **behavioral profile store**, the behavior of the user (such as location patterns, gait, movement dynamics, typing patterns, etc.). In addition, BAAs receive the locally captured from the user’s device behavioral profiles and store them for future **behavioral profile verification**.

Using the behavioral information, a BAA can determine whether a device currently behaves as it usually does. Depending on the result, it can certify to Service Providers or to the Identity Consolidator whether it believes the device is held by its legitimate user. This result, is delivered to the aforementioned entities by making use of the OpenID Connect specification (**Federated Logins**) and encapsulating the behavioral result as an attribute.

The BAAs also make use of the FIDO UAF specification in order to authenticate end-users before receiving data that will be used for the creation of a behavioral profile. This is achieved via the integration of OpenAM with the FIDO UAF Server (**FIDO enhanced OpenAM**). Specifically, when a user requires authentication to the BAA the BAA daemon acts as the Relying Party to OpenAM, which authenticates the end-users using the FIDO UAF specification.

To this end, BAAs can perform continuous behavioral authentication by constantly verifying end-users using their stored behavioral profiles. When a behavioral authentication fails, the BAAs inform the Identity Consolidator so that accounts can be locked by taking into consideration lock policies that were are configured by the user on the Identity Consolidator.

All the components of the BAAs are tunneled through the **gateSAFE** server so that adequate SSL protection is used for all the components. More details for gateSAFE can be found in Section 3.6.

2.4.1 Behavioral profiles database

This database will be responsible to store all of the user's behavioral profiles. These profiles will be created and stored by the input that will be provided by the behavioral capturing modules on BAAs and on user's devices. The database will provide input to the Behavioral profile verification module for the purposes of making sure that the captured profile matches the stored profile.

2.4.2 Behavioral profiles extraction

The BAAs will have a behavioral profile capture module. This module will be responsible for capturing user's behavior when requested either from Service Providers or from the Behavioral profile verification module.

This module should be able to capture local behaviors such as gait and typing patterns. There is an app on the user device that captures the behavior, temporarily stores it and sends it to the BAA (Telco, WEDIA, etc). The BAA stores this behavior to discern whether it changes over time. The user device securely erases the captured behavioral data so that they cannot be observed by an attacker who compromises the device after the behavior has been captured so he can use it to imitate the legitimate user's behavior.

The BAAs also capture the remote behavioral factors (e.g., location and internet traffic patterns) directly via their infrastructure so there is no need to have an on-device capturing module. Still there is a need for the BAA to have the proper remote behavior capturing module.

2.5 Identity Providers

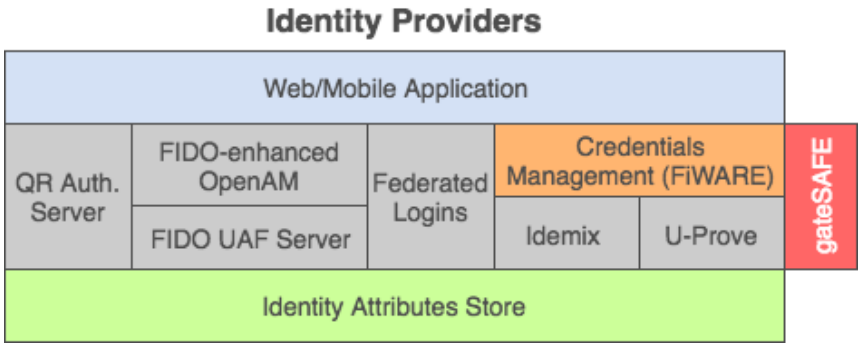


Figure 12 Identity Provider protocol stack view

In ReCRED, several Identity Providers (e.g., universities) are involved when required by the users or the Service Providers. Specifically, users may request credential issuance from the Identity Providers and Service Providers may request authentication of users. ID providers allow the storage of **identity attributes** of their users and the issuance of cryptographic credentials directly to the user’s device from the attributes using the **cryptographic credentials management** module that follow the **FiWARE** specification. The FiWARE interface enables the seamless use of the **Idemix** and **U-Prove** cryptographic stacks. Furthermore, the credentials management module is used for verification of Idemix and U-Prove credentials that result in the acquisition of trusted identity attributes. These attributes can be then transferred to other entities using the OpenID Connect specification that is realized by the deployment of the **FIDO-enhanced OpenAM** software stack. The credentials issued

from the Identity Providers are also transferred to the Identity Consolidator to store them so that they are backed up and accessible if the Identity Provider fails (only if the user consents to trust the Identity Consolidator).

The Service Providers or the ID Consolidator communicates with the Identity Providers using Federated Logins (as e.g., OpenID Connect [25] / OAuth [28]). Within the ReCRED architecture, the authentication of the end-users with the ID providers usually follow the FIDO UAF specification, by making use of the FIDO-enhanced OpenAM instance, and the identity attributes are then transferred to the Service Providers or to the ID Consolidator. Note that, by transferring ID attributes to the ID Consolidator, ReCRED is able to issue cryptographic credentials to the user device even if the ID Provider does not support exotic cryptographic protocols.

Furthermore, the Identity Providers have a **QR Authentication Server** that is able to verify information that was captured and send to the ID Provider from the QR client. This is particularly useful when a user wants to get access to a service using another device or browser (assuming that he is already authenticated). In this case, the user will scan a QR code using his device and he will send it to the QR authentication server on the ID providers. In that way, the ID provider will be able to authenticate the user and grant access to the service using QR codes authentication.

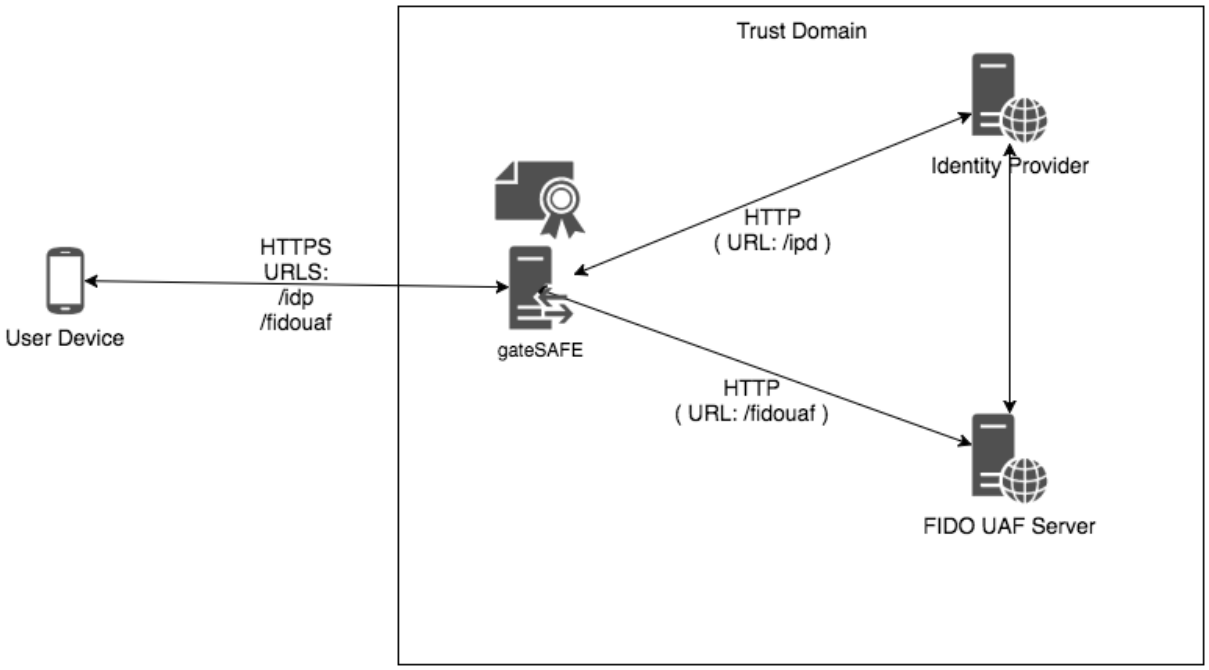
Similarly, to the other entities of the ReCRED architecture, all components are tunnelled through the **gateSAFE** product so that adequate SSL protection can be provided to all the components. More details for gateSAFE can be found in Section 3.6.

2.5.1 Identity Providers FIDO UAF Server

Users authenticates to the Identity Provider using the FIDO UAF protocol which implies the usage of the registration protocol as well. Just like in the case of BAA, the Identity Providers must have a FIDO UAF server deployed on their premises.

In the first step, the user registers with a specific Identity Provider and after a successful registration, the FIDO UAF Registration protocol is triggered. It is the Identity Provider’s responsibility to verify the user identity and the FIDO UAF Server relies on the IDP verification in order to successfully complete the registration step.

After a successful registration with the FIDO UAF Server, the user can securely authenticate with the IDP for subsequent operations (e.g. issuing U-Prove/Idemix cryptographic credentials). More details on how FIDO UAF server works can be found in Section 3.1.5.



Besides ensuring the TLS implementation, gateSAFE has also the role of protocol routing, with the responsibility to route the specific communication to the corresponding application server: Identity Provider or FIDO UAF Server. The routing decisions are taken based on the URL that the two applications are deployed on: As an example, the FIDO UAF server can be deployed on /fidouaf URL prefix, while the Identity Provider is deployed on /idp URL prefix.

2.5.2 QR Authentication Server

The QR Authentication component permits the device identity transfer from an authenticated smart-phone to a desktop. In ReCRED context this feature is an important one because a user can employ his already authenticated device (with ReCRED user-to-device and device-to-service methods) to access a protected resource by using a third-party desktop device.

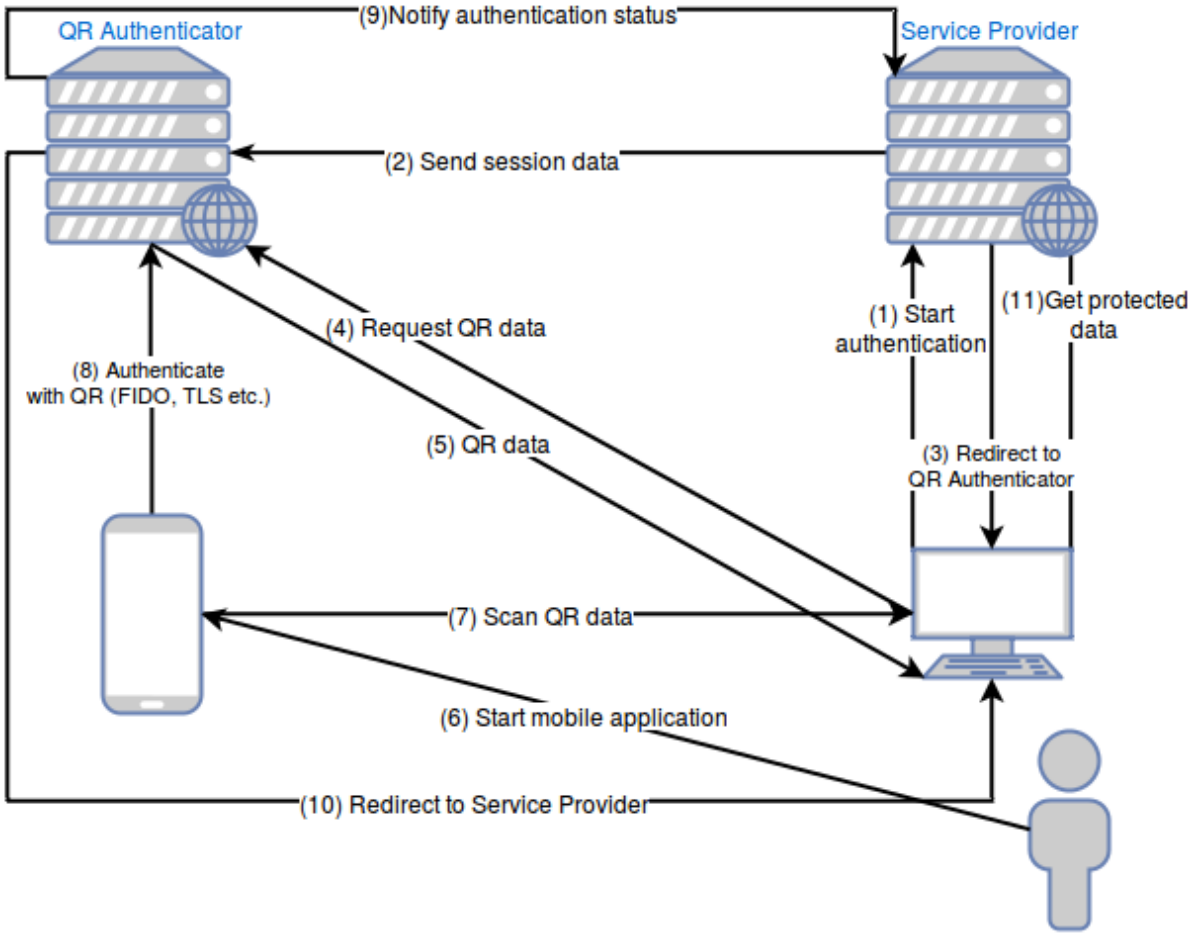


Figure 13 QR Authentication architecture

The QR Authentication process has a federated structure where the Service Provider delegates the authentication process according to a custom security policy. The QR Authentication steps along with the modules interaction are presented in the previous diagram. After the user opens the portal using the web browser from a desktop device, he initiates the QR Authentication process. The Service Provider generates a token associated with the current user session and sends a QR authentication request to the QR Authenticator module. The Service Provider request must contain a policy which describes the security parameters which will be used in the user authentication. The QR Authenticator verifies that the QR authentication request is from a trusted entity and generates an authentication message encoded in a QR code in order to be scanned by the mobile application. The user starts the mobile application and scans the QR code image displayed in the browser. The application decodes the QR code image and starts an authentication process with the QR Authenticator. The application could authenticate to the QR server by means of FIDO, mutual TLS or

other mechanisms. The QR Authenticator verifies the authentication message received from the mobile application and notifies the Service Provider regarding the authentication status. If the authentication succeeds, the user is redirected to the Service Provider and starts an authenticated communication session.

The authentication message structure is flexible because it permits the usage of various authentication scenarios and cryptographic algorithms. Also, the authentication message structure can be easily extended in order to fit authentication protocols with a complex structure.

For the QR code generation, the QR Authenticator will present distinct QR codes to the QR Device Client in order to avoid duplicate authentication data. The generated QR codes should contain a nonce generated according to the literature. Additionally, the QR code generation operation will employ a size limit so that the codes are not too big. In that way, the client can process it fast without causing QR code expiration. For the QR code transmission, the QR Authenticator needs to trust the source of the QR codes and to allow the QR Device Client to parse valid information and send authentication data to the same trusted server. For this reason, there should be a TLS channel between QR Server and QR Web Client. In order to mitigate a series of attacks the connection between the mobile device and the QR Server is also protected by a TLS channel. A TLS connection is assumed between the web client and the Service Provider.

2.6 Privacy-Preserving Attribute Based Access Control

One of ReCRED’s goal is to be able to offer a state-of-the-art Privacy-Preserving Attribute-based Access Control (PABAC) solution. The different components that comprise the ReCRED architecture were carefully designed in order to provide a PABAC solution on top of the OpenID Connect specification.

The goal of the PABAC solution is to allow Service Providers that do not know anything about cryptographic credentials to allow end-users to use already issued cryptographic credentials from various Identity Providers to get access to their service.

Below we give a brief description of the interfaces and components that are involved in ReCRED’s PABAC and how they interact in order to issue and verify cryptographic credentials for getting access to Service Providers.

- **Service Providers:** As described in the Service Providers dedicated section (Section 2.3), the Service Providers consist mainly of an XACML-compliant access control policies reasoning tool. This tool is responsible for receiving verified identity attributes from trusted entities (Identity Providers or Identity Consolidator) and deciding on whether access should be granted to users according to the defined policies and the received identity attributes. Note that the Service Providers entity will not be involved in verifying cryptographic credentials

(Idemix and U-Prove). Instead we delegate this functionality to trusted entities (Identity Providers or Identity Consolidator) that verify the credentials and deliver the identity attributes through the OpenID Connect specification.

- **User Device:** The user device will be responsible for the secure storage of the cryptographic credentials. Also, it will encompass the Idemix and U-Prove stacks that will responsible for releasing cryptographic credentials from the user device.
- **Identity Providers and Identity Consolidator:** These entities are considered as trusted entities and are responsible for the issuance and verification of cryptographic credentials. In a typical scenario, a user will visit a particular Service Provider and then he will be redirected to an Identity Provider or the ID Consolidator in order to verify some identity attributes. Subsequently, the user will present its cryptographic credentials to the IdP or the IDC, which will responsible for verifying the credentials using the underlying Idemix or U-Prove stacks. After verifying the credentials, the IdP or IDC will provide the verified identity attributes to the Service Providers via the OpenID Connect specification.

  From an implementation perspective, to achieve the aforementioned scenario in the IdPs and the IDC we will integrate the Idemix and U-Prove stacks within the OpenID Connect Provider implementation (OpenAM). Specifically, we will implement a custom authentication module that will allow the seamless use of the two underlying cryptographic stacks and will be responsible for verifying presented PABAC credentials and releasing the appropriate identity attributes to Service Providers.

Figure 14 presents an overview of the PABAC components and how the cryptographic credentials verification is achieved.

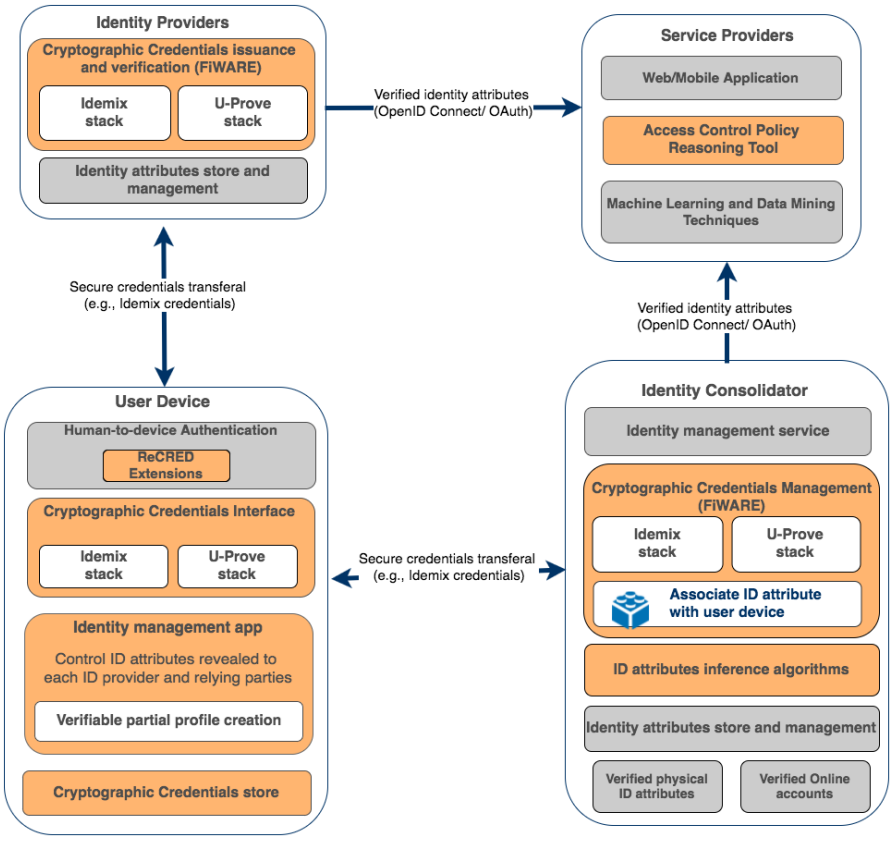


Figure 14: Attribute-Based Access-Control component view

3 Protocols and Interfaces

3.1 FIDO Integration in the ReCRED Architecture

ReCRED integrates the FIDO Universal Authentication Framework [12] in its protocol stack and preserves user privacy aspects like: untraceability and unlinkability. Users disclose minimally information about their owned attributes in the process of registration by using an attribute-based authentication scheme. At the same time the architecture preserves the compatibility with the FIDO UAF protocol.

FIDO UAF was created to allow Identity Providers to offer user authentication based on a password-less mechanism (e.g., fingerprint recognition). ReCRED extends the authentication mechanism, and takes advantage of the attribute-based encryption schemas, backed by a trusted execution environment contained in the user mobile device.

On the user device, there are two main components that implement the FIDO UAF Protocol:

- FIDO UAF Client, as it is referred in the FIDO documentation

- FIDO Authenticator, as it is referred in the FIDO UAF Specifications

On the Identity Provider side, the main component that implements the FIDO UAF protocol is called FIDO Server. In order to adapt to FIDO specifications, the ReCRED architecture introduces another component called Protocol Adapter that takes care of the protocol specific responsibilities.

There are three main protocols involved in the Architecture that are compliant with FIDO specifications as well:

- Registration Protocol, which has the role of uniquely associating a user with a specific service preserving the user privacy
- Authentication Protocol, that ensures that the user authenticates in a secure way to a specific service. Authentication can be direct or federated
- Federated Authentication Protocol, that moves the specific authentication steps from the Service Provider to a Federated Identity Provider
- De-registration Protocol, which removes the association of a user with the specified service.

The following diagram shows the architecture components that are involved in the FIDO protocols implementation.

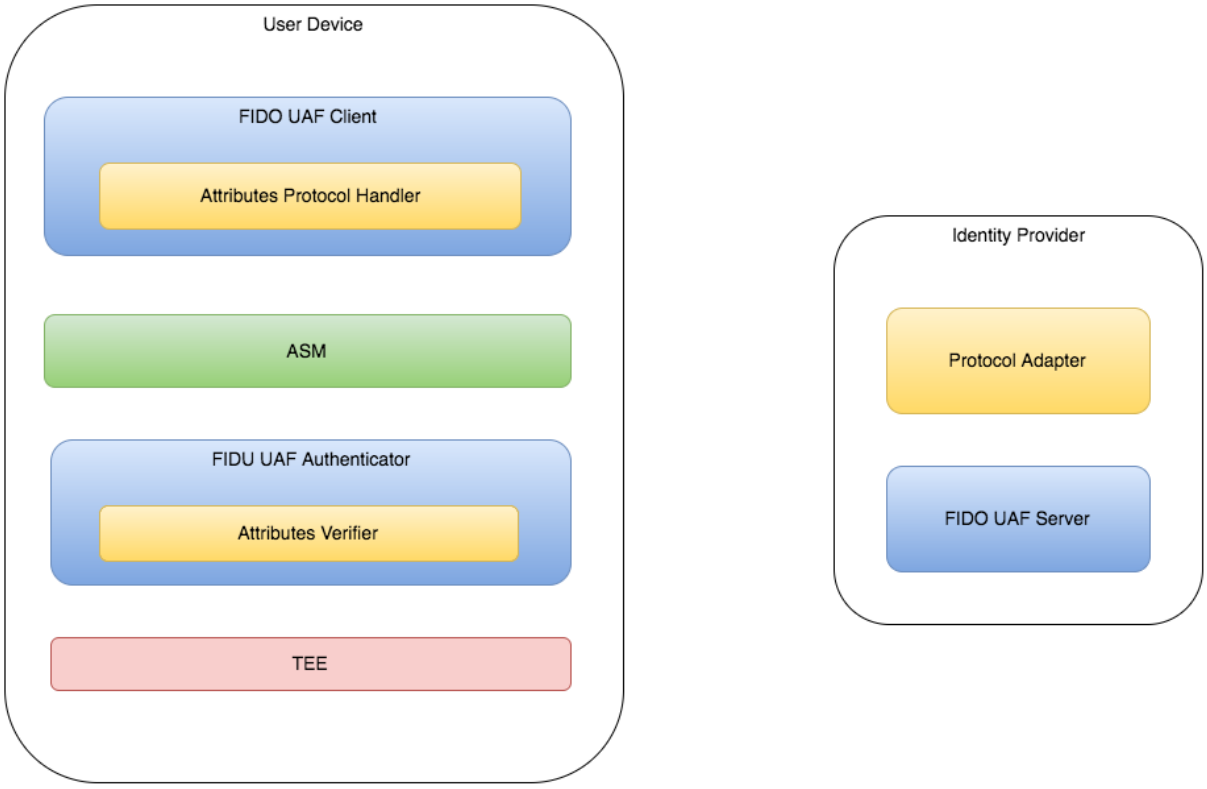


Figure 15 FIDO Components

- The FIDO UAF Client implements the client side of FIDO UAF and transports the underlying protocol specific authentication and registration messages. It is responsible for:

- communicating with the FIDO server, not directly but through a user agent (an application that runs on the user device that can be either a mobile application or a standard internet browser). Therefore, the FIDO client will interface with the user agent and will deliver and receive messages from it. The user agent forwards FIDO protocol specific messages to the FIDO server through the relying party web application. The user agent contains a FIDO-specific implementation (e.g. a browser plugin) that handles the FIDO messages.
- communicating with the FIDO UAF Authenticator, exchanging protocol specific cryptographic messages (e.g. U-Prove or Idemix). The FIDO Client communicates with the FIDO Authenticator using a standardized interface: the Authenticator Specific Module, which provides a uniform interface between the client and the hardware.

- The FIDO UAF Server implements the server side of the FIDO UAF protocols and is responsible for:
  - interacting with the Web Application, through which it communicates UAF protocol messages to the FIDO UAF client
  - evaluating client authentication responses to determine their validity
  - managing the association of FIDO UAF Authenticators with user accounts in the Web Application
  - interacting with the Protocol Adapter to obtain attribute-based specific messages
- The Protocol Adapter is a specific component that extends the FIDO UAF authentication protocol by adding attribute-based specific messages interpretation. The Protocol Adapter is responsible for implementing attribute-based authentication.
- The Authenticator Specific Module (ASM) is an abstraction layer that provides a uniform API between the authenticator hardware and the FIDO Client. The FIDO Client uses the ASM API calls in order to delegate cryptographic operations to the Authenticator module. Also, the ASM interface allows the use of different authenticator modules and ensures the compatibility between the authenticator and the client.
- The FIDO UAF Authenticator is the secure element connected to all or part of the user device (mobile phone) that is responsible for the creation of the cryptographic key material, storage and secure usage of the private keys used in the authentication protocol. In case the device contains and uses a Trust Execution Environment (e.g., ARM Trust Zone), the authenticator can use this safe storage and execution memory/CPU to obtain a higher level of security. How the FIDO Authenticator interacts with the TEE is specific to each implementation and is heavily dependent on the device hardware.

3.1.1 Registration Protocol

The registration process is very specific to the underlying protocol (e.g. in the context of attribute based access control, the registration protocol can create attribute information which is specific to the Identity Provider: token information and/or private key material that is associated only with a specific Identity Provider).

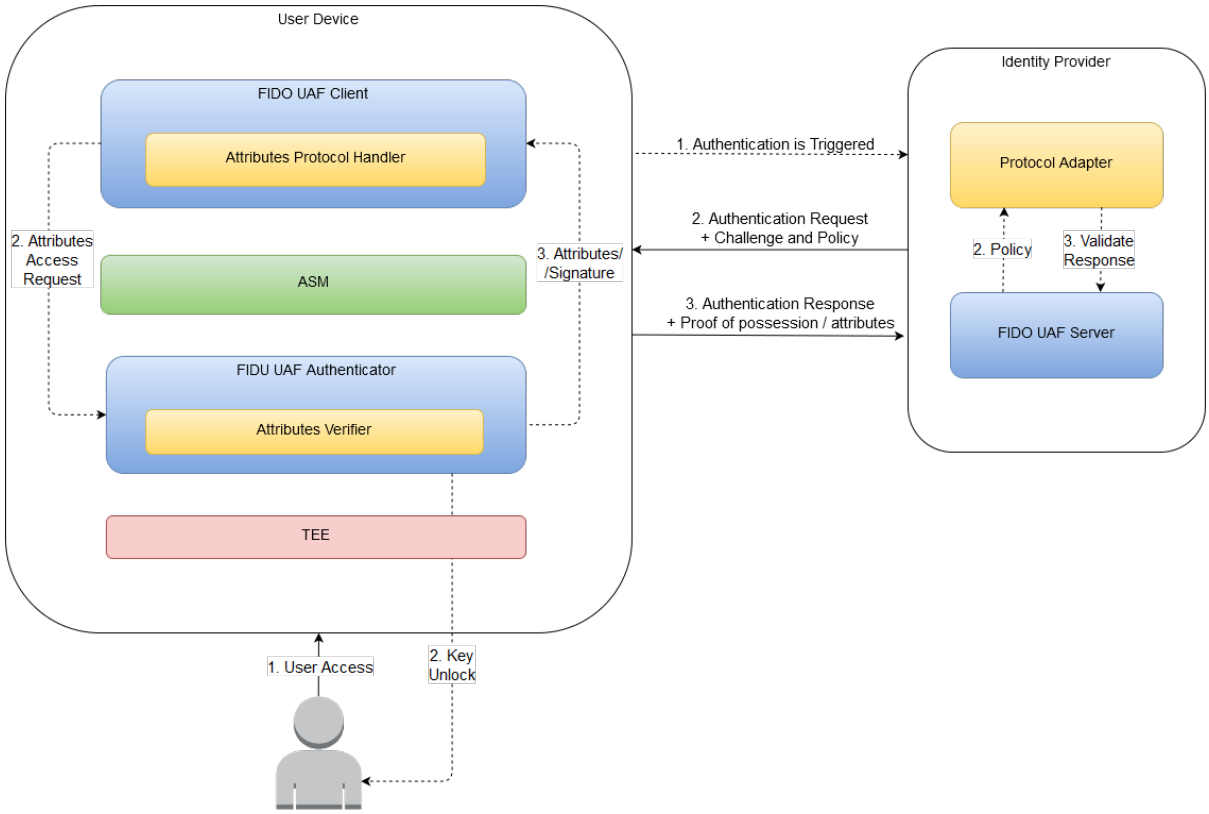
Nevertheless, the Identity Provider can choose to create a user pseudo account based on a temporary identifier that it and the user decide upon and use that 'pseudo' account in order to remember the details that the user chooses to remember.

3.1.2 Authentication Protocol

In the authentication process, the website verifies that the user is allowed to access a restricted resource. The entities involved in the user verification are: the FIDO Server and the Protocol Adapter. The Identity Provider issues an authentication request message.

Generically, the authentication protocol follows the rule of a simple request-response message exchange, where server sends a challenge and an accepted policy and the user responds with a proof of private key possession.

On the client side, the client receives the authentication request and verifies the server and the policy. Because the server is not a subject whose privacy should be preserved, it can have a public ID vouched by a third party trusted entity (e.g., an X509 certificate). Establishing the server’s identity is not part of this protocol and the protocol relies on other protocols to validate it (e.g. SSL)



3.1.3 Federated Authentication Protocol

Federated Authentication protocols complement FIDO UAF in the ReCRED architecture. The basic authentication concepts that FIDO standardizes are still in place, but in this case the authentication happens between the client and a third party that the Service Provider chooses to trust.

Figure 16 FIDO Authentication Protocol

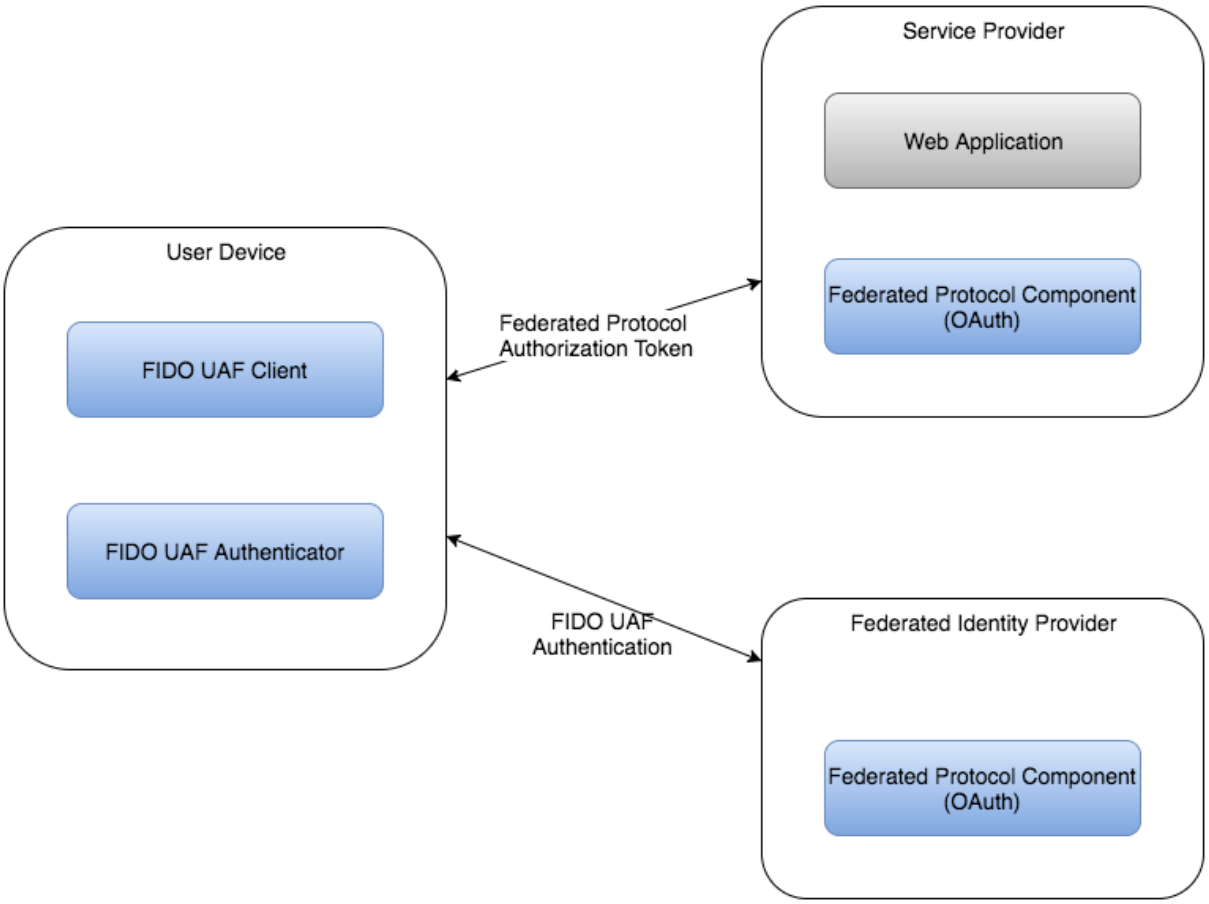


Figure 17 FIDO and Federated Authentication Protocols

There is another protocol involved in the authentication mechanism: a protocol that allows the Federated Identity Provider to talk to the Service Provider and establish whether a specific client identified with a specific ID is authenticated or not (and authorized as well if the protocol allows it). To achieve this, the Federated Identity Provider issues a verification token, which is usually a short-lived token that can be used only once and that arrives at the Service Provider through the User Device (user agent). Usually, the user agent chooses to exchange this token with an access token (and a refresh token sometimes) and from that moment the verification token becomes unusable. The user agent will further use the access token to access the Service Provider resources.

Before the authentication process, there is an enrolment process that takes place between the Service Provider and the Federated Identity Provider: a process in which the two entities establish a trust relationship. Based on this established trust, the two entities will exchange IDs that the user agent will use in the authentication process.

3.1.4 De-registration Protocol

A de-registration process can occur if a registration process has happened previously. Although FIDO provides such a mechanism, it is tightly dependent on the underlying protocol. In general, if the client created private key material or attribute data specific to a particular Service Provider, the client will delete this data from its space (including the Authenticator).

3.1.5 FIDO UAF Server

3.1.5.1 Registration/Authentication

The FIDO UAF Server allows user registration/authentication based on a policy that can define generic authenticators, target specific ones, or specify a list of disallowed authenticators. After receiving a policy from the server, the FIDO UAF Client selects suitable authenticators available on the system and presents to the user this list, otherwise it stops the registration/authentication.

3.1.5.2 Policy

The server policy is a member of every registration/authentication request sent to the FIDO UAF Client.

It is a structure containing two **MatchCriteria** array members. The first member is a two-dimensional array interpreted as a list of sets, the list elements (sets) can be viewed as alternatives for allowed authenticators. The second array member of **MatchCriteria** objects represents the authenticators which will be rejected by the server.

The **MatchCriteria** structure mirrors to a certain degree the FIDO UAF Authenticator Metadata Statement which contains information about how the authenticator enrol and authenticates users to the device, algorithms used to compute signatures and encodings used to transport the public keys to the server.

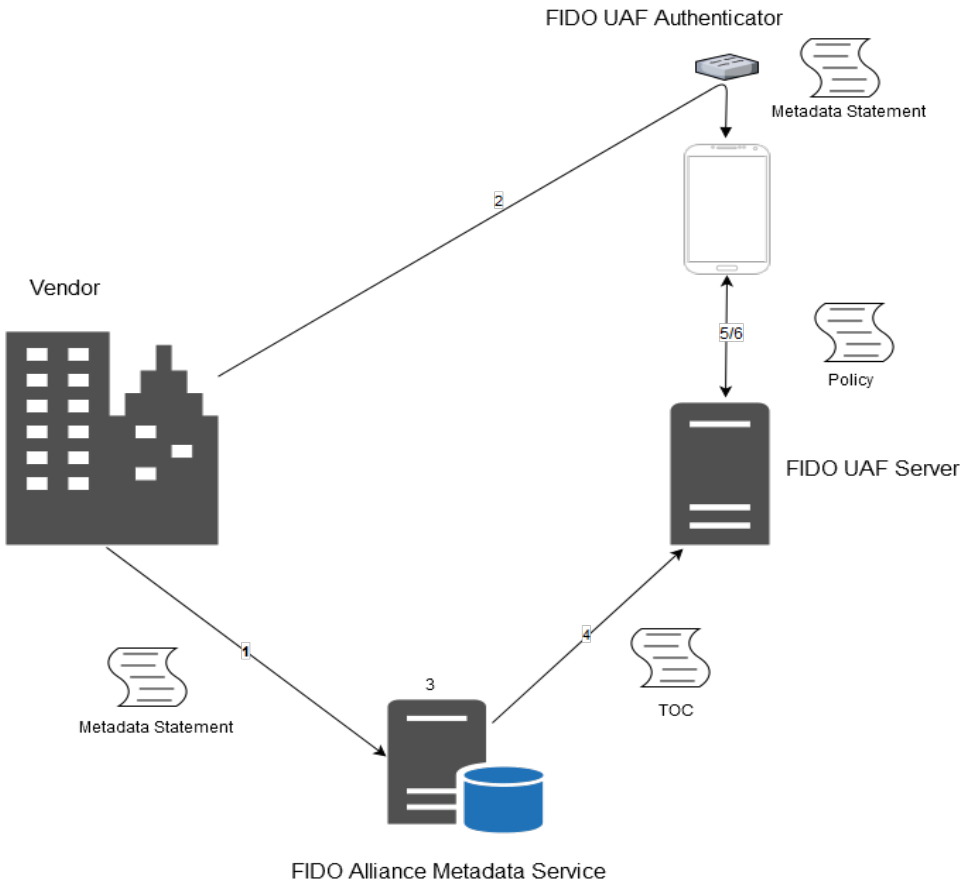
3.1.5.3 FIDO UAF Authenticator Metadata Statement

The metadata statement contains the “Trust Anchor” required to validate the attestation object and characteristics about the authenticator implementation such as user enrol and verification method (pin, password, biometric, pattern, etc.), relevant accuracy/complexity aspects for the verification methods, authentication algorithms, key encodings, root attestation certificates and various information such as a short description, vendor id, authenticator id and version, transaction display characteristics, etc.

Based on the metadata statement the FIDO UAF Server creates the registration/authentication policy and assigns a risk or trust score to authenticators.

3.1.5.4 Metadata Statement acquisition

Every FIDO UAF Authenticator vendor submits the metadata statement to the FIDO Alliance which publishes the statement to a Metadata TOC (table of contents) file accessible through the FIDO Alliance Metadata Service (MDS).



Every time the vendor obtains a FIDO Certification for an authenticator the following process occurs:

1 The vendor supplies the metadata statement to the metadata service;

2 The user acquires an authenticator from the vendor which contains the metadata statement;

3 The metadata service appends the statement to the TOC and signs it;

4 The server downloads the TOC file, verifies the signature and creates a registration/authentication policy based on it;

5 The server registers/authenticates users based on the policy defined from the TOC file.

FIDO Alliance supplies the TOC file as a JWT (Javascript Web Token), the root certificate (used to verify the JWT certificate chain) and the certificate revocation information (CRL) through HTTPS (https://mds.fidoalliance.org). The TOC file specifies the next date when the server should acquire the updated version of the file.

The main purpose of the Metadata Service is to keep the FIDO UAF Server up to date with the latest authenticators and their security updates, providing interoperability with newer client stack protocol implementations.

The metadata TOC file contains three sections, first is the certificate used to sign the file, the second is the metadata TOC entries – which contain information about authenticators, links to the vendor specified metadata statement, FIDO certification status, vulnerabilities (if the authenticator has them) and the time when the vendor will update the metadata statement. The last section of the file is the signature, computed over the previous sections.

This metadata acquisition method represents a dynamic approach, where the server creates the registration/authentication policy based on the newest authenticators on the market, allowing the user to authenticate with his preferred authenticator. On the other hand, if only a well known set of authenticators with a certain implementation will be used to authenticate with the server, the policy can be hard coded.

3.1.5.5 Registration/Authentication Assertions

Assertions contain information encoded in **TLV** form, generated by the FIDO UAF Authenticator as a response to the registration/authentication requests sent by the server, they are passed to the FIDO UAF ASM, which sends them to the client, finally arriving at the server for further processing.

Registration Assertion

The most important structure in the registration assertion is the **KeyRegistrationData** (KRD) which contains the public key, it`s id, plus the registration and signature counters – indicating how many register and sign operations the authenticator has performed (useful for detecting cloned authenticators).

Another structure in the assertion is the attestation one which can be of the type ATTESTATION_BASIC_FULL or ATTESTATION_BASIC_SURROGATE.

In the basic full attestation, the whole KRD structure is signed with the attestation key, attesting to the server that the public key was generated with a trusted authenticator, the attestation certificate is also sent to be verified with the attestation root certificate contained in the metadata statement on the server. In the basic surrogate attestation, the whole KRD structure is signed is signed with the private counterpart of the key, assuring the server authenticator has possession of the private key.

Authentication Assertion

The SignedDataStructure structure in the assertion contains the id of the key used to sign the challenge parameter, the signature counter – used by the server to detect cloned authenticators and the signature computed with private key counterpart of the key with the preceding id. The server receives this structure, finds the public key with the given id and verifies the signature.

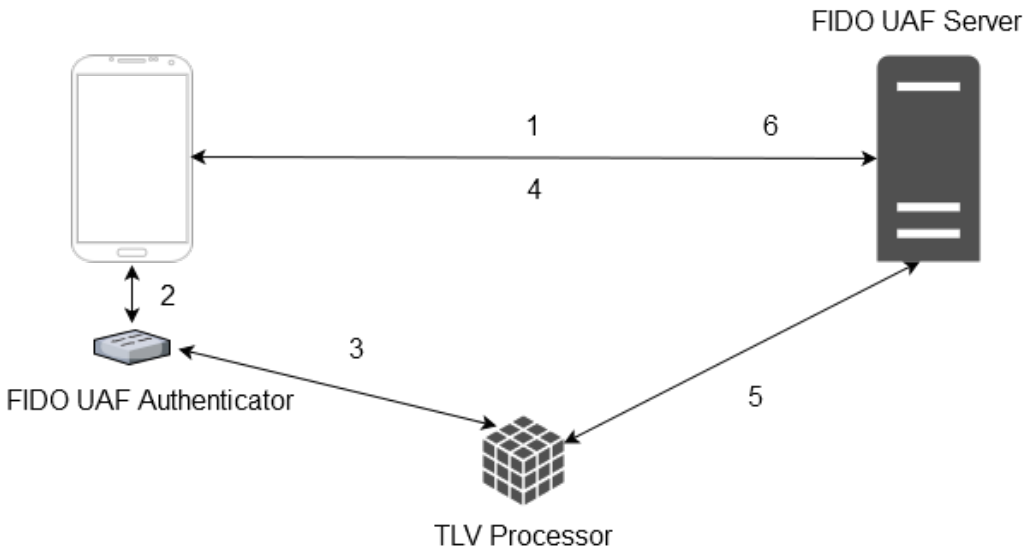
3.1.5.6 TLV

FIDO UAF Authenticator communication with the server is made by using TLV encoding “Tag-Length-Value”. TLV encoded data has the following structure:

2 bytes	2 bytes	Length bytes
Tag	Length in bytes	Data

(table from fido-uaf-authnr-cmds-v1.0-ps-20141208.pdf page 6)

Tags contain important protocol data and structures such as public keys, attestation structures, signatures, certificates, counters, etc.



Both FIDO UAF Authenticator and server share the same TLV processor code base to encode and decode registration/authentication data:

1. FIDO UAF Server sends an registration/authentication request to the client;
2. The client, through the ASM module, commands the authenticator to create a response based on the request;
3. The authenticator encodes the registration/authentication assertion structure with the help of the TLV processor;
4. The client sends the authenticator assertion to the server so it can process it;
5. After the TLV processor decodes the TLV data, the server verifies the attestation;
6. The server informs the client about the attestation verification status and if the registration/authentication process finalized correctly.

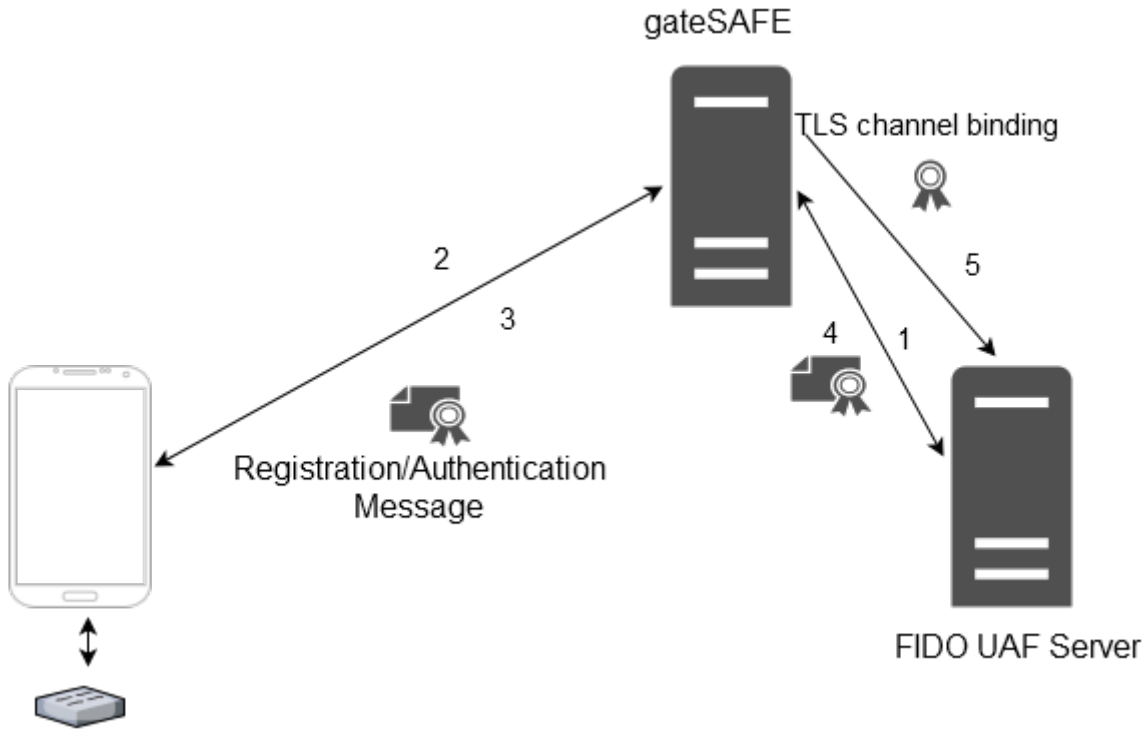
3.1.5.7 TLS Channel binding

TLS channel binding is used by the FIDO UAF Server to detect and prevent MITM attacks. If the FIDO UAF Client has access to TLS channel information, it can send it to the server in an registration/authentication response.

The channel binding methods available to the FIDO specifications are:

- TLS ChannelID – base64url-encoded hash of the TLS server certificate;
- serverEndPoint – base64url-encoded TLS channel **Finished** structure [RFC5929]
- tlvServerCertificate – base64url-encoded, DER-encoded TLS server certificate;
- tlsUnique – base64url-encoded serialized [RFC4627] **JwkKey** structure containing the public key minted by the client TLS stack.

From the gateSAFE SSL Termination feature, the FIDO UAF Server can extract one of the channel binding structures described above and verify that they are in fact the same with the ones received from the client:



1. The server sends through gateSAFE the registration/authentication request;
2. gateSAFE sends the request to the client;
3. The client sends the response with TLS channel binding included;
4. The server receives the response with channel binding from gateSAFE;
5. The server verifies the channel binding information from the response with information retrieved from gateSAFE.

3.2 OpenID Connect/OAuth

3.2.1 Introduction

Identity delegation is an act whereby an entity delegates his or her authority to use identity information to another entity. Some emerging technical specifications provide schemes for exchanging identity information using a short string token. One of the most prominent technologies that use tokens is the Security Assertion Markup Language (SAML) [26] which specifies a scheme for exchanging identity information using an artifact, which is a reference to an assertion containing the information. However, SAML does not deal with access control schemes and the artifacts are not specifically designed for identity delegation, while it is not considered optimal for mobile devices (see below).

Another specification for identity delegation is OAuth 1.0 [27], which is an open standard for authorization. The second version of OAuth 2.0 [28] is the next evolution of the OAuth protocol and is not backwards compatible with OAuth 1.0. OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows for Web applications, desktop applications, mobile phones, and living room devices. The main difference between OAuth 1.0 and 2.0 is that OAuth 1.0 is a standard protocol for identity delegation, whereas 2.0 is a highly extensible framework. The protocol itself has been described as inherently insecure by security experts and a primary contributor to the specification stated that implementation mistakes are almost inevitable. OAuth 2.0 does not support signature, encryption, channel binding, or client verification. It relies completely on TLS for some degree of confidentiality and server authentication.

3.2.2 OpenID Connect

OpenID Connect [25] is a simple JSON/REST-based identity protocol built on top of the OAuth 2.0 and JWT (JSON Web Token) [19] family of protocols. In particular, OpenID Connect is a simple identity layer on top of the OAuth 2.0 protocol, which allows computing clients to verify the identity of an end-user based on the authentication performed by an authorization server, as well as to obtain basic profile information about the end-user in an interoperable and REST-like manner.

The goal was to develop a single protocol that is:

- Mobile friendly, i.e. supporting native applications and the generally restricted bandwidth and capabilities of feature phones.
- A single flow integrating API authorization and user authentication, seamlessly.
- Lightweight to implement for applications and relying party websites, replacing proprietary last mile integrations.

The major actors and their role in OpenID Connect are:

- A user (also known as Resource Owner) is a person who accesses restricted resources provided by other entities. A user has his or her intimate users' contact addresses (e.g., account identifiers or e-mail addresses) through which they can interact with each other. In this model, there are two types of users: delegators and delegatees. A delegator is a user that has privileges to access his or her identity information and to delegate the privileges. A

delegatee is a user that is provided privileges by a delegator to access the delegator's identity information. A delegator and a delegatee have a prior trust relationship and share their contact addresses.

- A Service Provider (SP) (also known as Relying Party or Client) is an entity that provides access to restricted resources managed by its site with authorized users. An SP supports the delegation of resources containing personal information. This means that an SP may grant a delegatee's access to a delegator's resource on the basis of its security policies if an SP verifies the appropriateness of the delegated access request by the delegatee.
- An Identity Provider (IdP) is an authentication authority that authenticates users by means of particular authentication methods and produces an assertion stipulating the completion of the authentication event or the correctness of personal attribute information. In this model, an IdP has an additional Delegation Mediating (DM) capability that supervises the delegation authorization of a delegator to a delegatee and conveys its information to a corresponding SP.

Please note that the aforementioned roles can be switched according to the use case. For instance, in one use case the Identity Consolidator may act as the Identity Provider and in another use case may act as a Relying Party to other Identity Providers.

Part of OpenID Connect design philosophy was to push complexity to the OpenID Connect Identity Provider (IdP) where possible. This is in contrast with SAML, where the burden of supporting the protocol is much more evenly shared between the identity provider and the service provider. Because of this asymmetry in support complexity, OpenID Connect has much greater applicability to native mobile applications — which are necessarily far more constrained than server implementations.

One of the key areas of simplification from SAML was migrating from XML to JSON, which has wide support in all modern programming environments — being the data representation format used by JavaScript in all modern Web browsers. The JWT format, including its methods for Signing and Encryption (JOSE), allows for more compact security tokens than is possible by using XML (and realized by SAML). Small security tokens are required to fit the URL and header size constraints in mobile environments. The JWT format also avoids the problems that XML signing and encryption have been plagued with over their history.

In the following figure a comparison between SAML 2.0 & OpenID Connect is shown [1].

	SAML 2.0	OpenID Connect
Service Provider	• Service Provider libraries	• Relying Party libraries
Identity Provider	• Identity Provider libraries	• OpenID Provider libraries
Attribute Provider	• Attribute provider provides further detail to enrich SAML Assertion • Requires further step to populate assertion with user attributes	• OpenID Provider • The UserInfo Endpoint returns Claims about the End-User
Attributes	• SAML Attributes	• OpenID Connect Scopes (groups of attributes
Discovery Service	• No • Requires pre-agreed metadata	• Single discovery service for RP allowing sites & apps can “validate” your users
Privacy	• Yes	• Yes, JSON Object Signing and Encryption (JOSE)
Signing	• Yes	• Yes, JSON Web Token (JWT)
Mobile Apps	• No, SAML Web Profile for Web Browser only	• Both web browser & mobile apps
Support for SSO	• Web SSO only	• Yes
Form Rendering	• Both SP & IdP	• Normally Identity Provider

Figure 18 SAML 2.0 & OpenID Connect comparison

3.2.3 Protocol Overview of OpenID Connect

The OpenID Connect protocol, in abstract, follows the following steps.

1. The sequence starts with the end-user making some request through the client (step 1). The latter redirects (step 2) the end-user's browser to a URL at the identity provider.
2. The identity provider authenticates the End-User and obtains authorization (step 3).
3. The identity provider redirects the End-user to the client with an authorization code (step 4).
4. The client sends a direct POST request (step 5) to the identity provider along-with an authorization code.
5. The identity provider responds with an ID Token and usually an Access Token (step 6). The ID token is a signed data structure that contains authenticated user attributes, encoded as a JSON Web Token (JWT). As well as identifying the end-user, it also identifies the token issuer and, crucially, the client to which the token was issued.
6. The RP can send optionally a request with the Access Token to the User Info Endpoint (step 7).
7. The User Info Endpoint returns Claims (step 8) about the End-User (e.g., email, gender, photos, etc.).

At the end of the protocol, the user has been authenticated to the client and to the identity provider, while it has also authorized the client to access its attributes using the access token obtained from the identity provider.

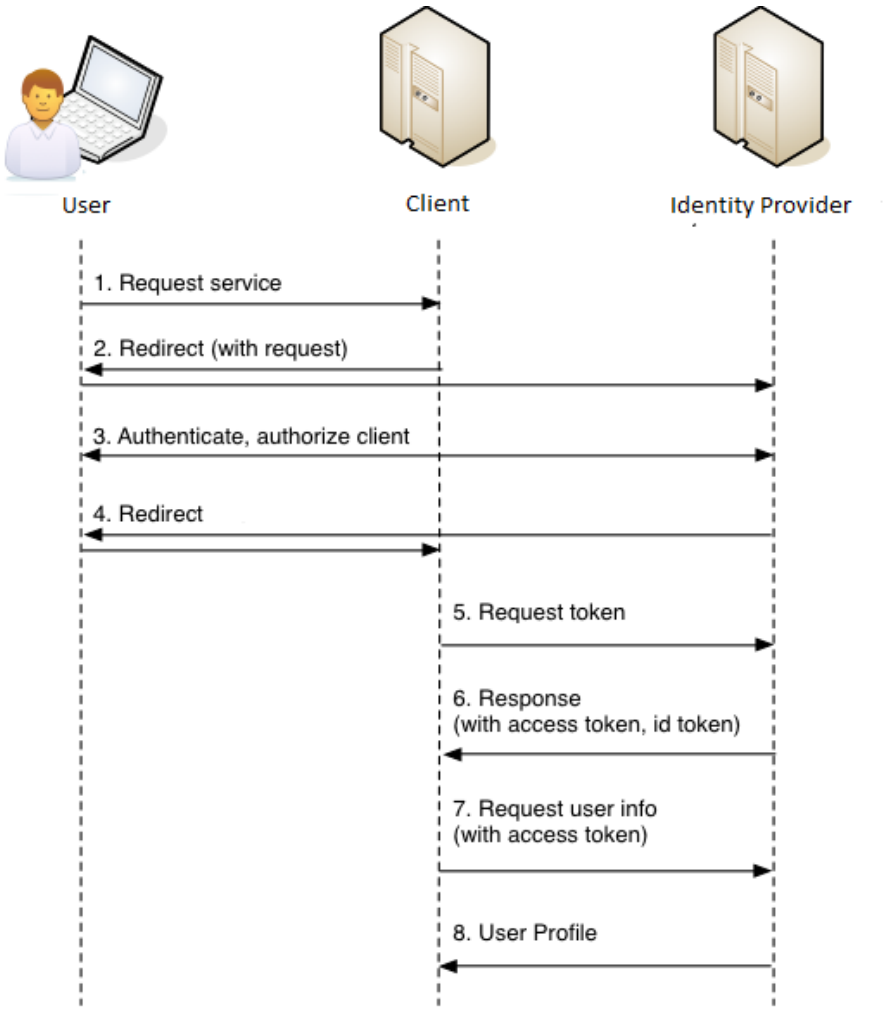


Figure 19: OpenID Connect protocol execution flow

3.2.4 OpenAM

To exploit the OpenID Connect Provider features, ReCRED will make use of the OpenAM software. This is a web-based open-source solution that provides authentication, authorization, entitlement, and federation services. OpenAM supports 25 authentication methods and offers the flexibility to create custom authentication modules based on the JAAS (Java Authentication and Authorization Service) open standard. In addition to that, OpenAM’s federation services allow federated members to securely share identity information with each other. During the ReCRED project, we will extend OpenAM’s to match ReCRED’s specific goals. Some extensions include:

1. Custom authentication module for performing FIDO UAF authentication with the OpenAM software. This module is currently in-use for the Wi-Fi pilot.
2. Custom authentication module that will act as cryptographic credentials verification process. The users should be able to present credentials and the module will be able to verify the credentials using the FiWARE interface.

3. Integration with the identity Consolidator's Identity Repository. This will include the implementation of an interface that will call the Storage API in order to retrieve data from the Identity Repository.

3.3 Mobile connect

This is the GSMA Personal Data programme focused on positioning mobile network operators as trusted providers of identity services to 3rd party service providers. The programme identifies a set of propositions (including authentication, validated identity, enhanced profile, attribute brokerage) that collectively are referred to as Mobile Connect. Discovery is the process of finding out the IDP (home operator for the user) to which the user belongs and getting the technical information to connect to the IDP for the Identity Services. Mobile Connect is based on the OpenID Connect / OAuth2 standards, where MNOs are principally responsible to authenticate end-users and to provide users’ attributes to the service providers.

One of the key goals of the ReCRED project is to have an implementation of Mobile Connect and FIDO within ReCRED platform. Furthermore, the Identity Consolidator acts as a Mobile Connect proxy and offer Mobile Connect as a Service to Service Providers. To achieve this, Identity Consolidator’s or MNO’s authentication management engine needs to be configured as OIDC Provider and should also provide an interface in form of a custom authentication module to facilitate in the overall process of the FIDO authentication.

One of the main benefits is that via this approach, it is possible to get LOA4 attributes from end users in the identity consolidator; making it a very powerful tool for disclosing attributes.

Also, by adding the mobile connect discovery solution to the Identity Consolidator, service providers can integrate very easily with the identity consolidator as well as with mobile connect in one go.

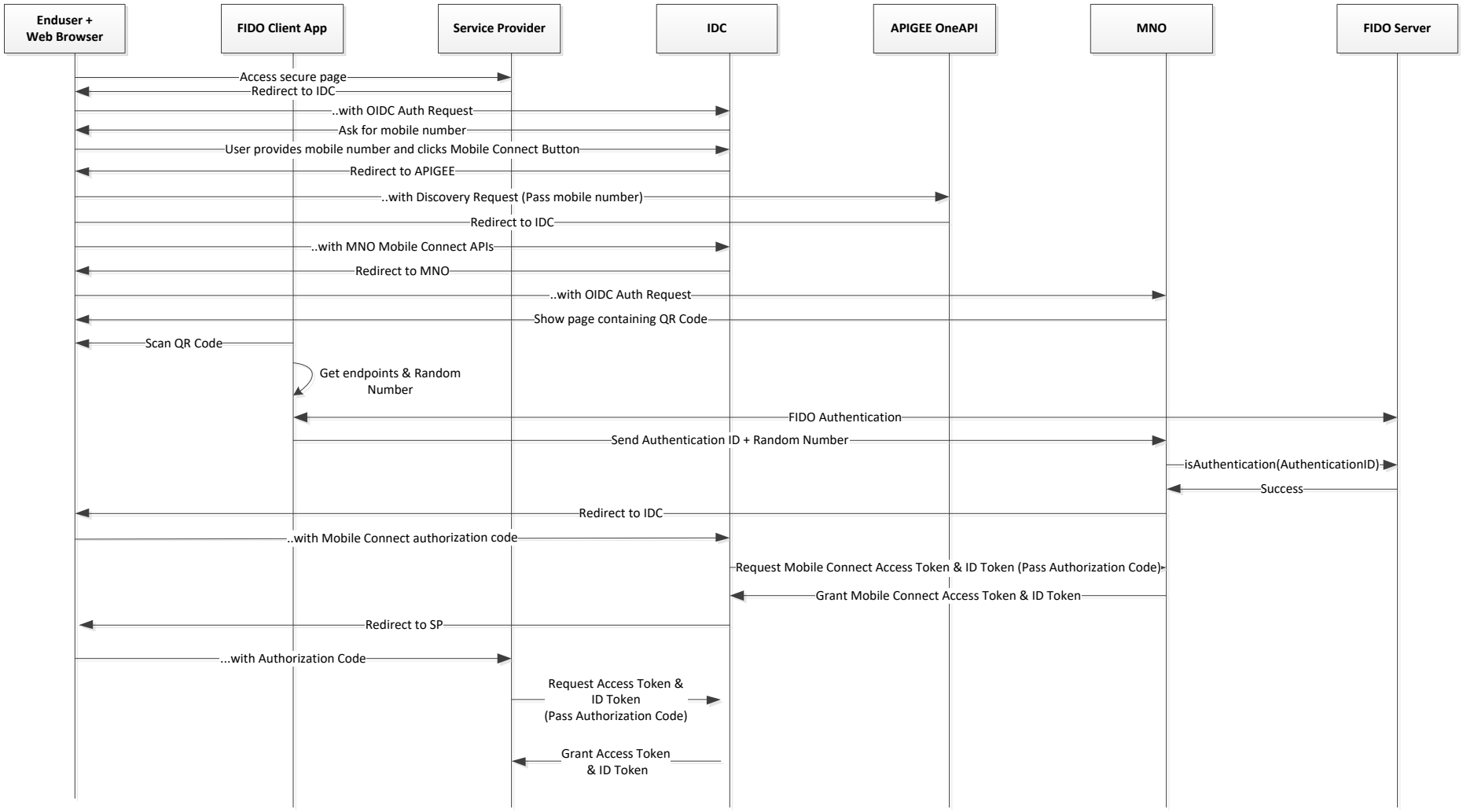
The following diagram shows a flow where a user would authenticate via the identity consolidator with a mobile connect MNO.

The diagram describes the low level calls. On a high level:

1. The Service Provider sends an authentication request to the IDC
2. The Identity Consolidator requests the user’s phone number
3. The Identity Consolidator sends a discovery request to APIGEE OneAPI
4. APIGEE OneAPI looks up the right MNO for the user.
5. The user is now redirected to the right MNO.
6. Depending on the authenticators implemented on the MNO and the requested level of authentication, the MNO now authenticates the user.
7. The user is now sent back to the Identity Consolidator with an authorization token that is used to retrieve an access token to be used between the identity consolidator and the MNO.

8. After receiving this information, the identity consolidator generates a new authorization code which in turn is used to get an access token to be used between the Identity Consolidator and the Service Provider.

In this use case, we observe that the Identity Consolidator both act as the Identity Provider and the Relying Party. It acts as the Identity Provider for the Service Provider and as a Relying Party to the MNO. This use-case demonstrates the flexibility of the ReCRED architecture.


Enduser + Web Browser
FIDO Client App
Service Provider
IDC
APIGEE OneAPI
MNO
FIDO Server
Access secure page
Redirect to IDC
..with OIDC Auth Request
Ask for mobile number
User provides mobile number and clicks Mobile Connect Button
Redirect to APIGEE
..with Discovery Request (Pass mobile number)
Redirect to IDC
..with MNO Mobile Connect APIs
Redirect to MNO
..with OIDC Auth Request
Show page containing QR Code
Scan QR Code
Get endpoints & Random Number
FIDO Authentication
Send Authentication ID + Random Number
isAuthentication(AuthenticationID)
Success
Redirect to IDC
..with Mobile Connect authorization code
Request Mobile Connect Access Token & ID Token (Pass Authorization Code)
Grant Mobile Connect Access Token & ID Token
Redirect to SP
...with Authorization Code
Request Access Token & ID Token (Pass Authorization Code)
Grant Access Token & ID Token

3.4 Cryptographic Credential Stacks

3.4.1 Idemix

3.4.1.1 Idemix Description & Overview

Idemix [5][10][4] is an identity management system based on anonymous credentials and zero-knowledge protocols. Parties involved in the Idemix system can play the roles of issuers, recipients, provers and verifiers. The issuer represents the authority demanded to the issuance of credentials (for which he is responsible) to a recipient through an issuance protocol that results in the recipient owning a credential. When a proof of possession of credential is required by a verifier, the credential owner (the recipient that obtained the credential issued by the authority) acts in the role of a prover towards the verifier. An Idemix credential consists of a set of attribute values as well as cryptographic information that allows the owner of the credential to create the proof of possession. As shown in Figure 20 and Figure 21, the components of the ReCRED reference architecture are easily mapped into the above described roles of the Idemix architecture. Note that, the Credential Management (CM) module deployed in the Identity Consolidator acts as a special instance of an Identity Provider. Indeed, it is in charge of issuing credentials based on the consolidation of identities and it can act as an Identity Provider proxy that releases credentials on behalf of Identity Providers that does not support cryptographic credentials.

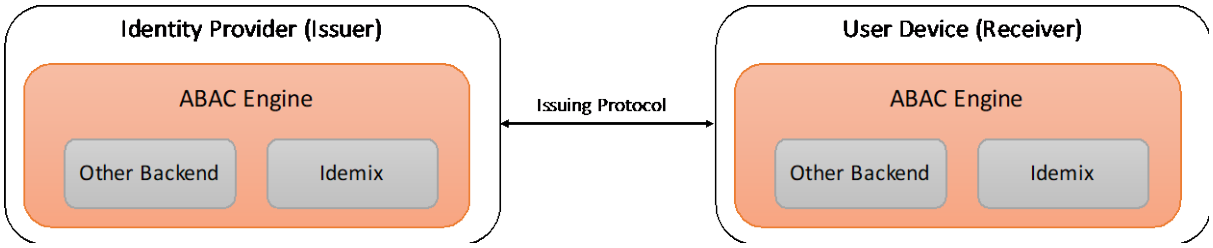


Figure 20 Mapping of Identity Provider and User Device respectively to an Issuer and a Receiver running the Issuing Protocol to allow the release of credential

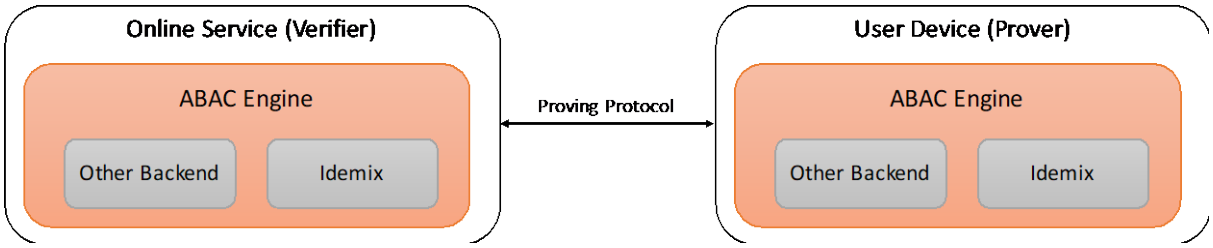
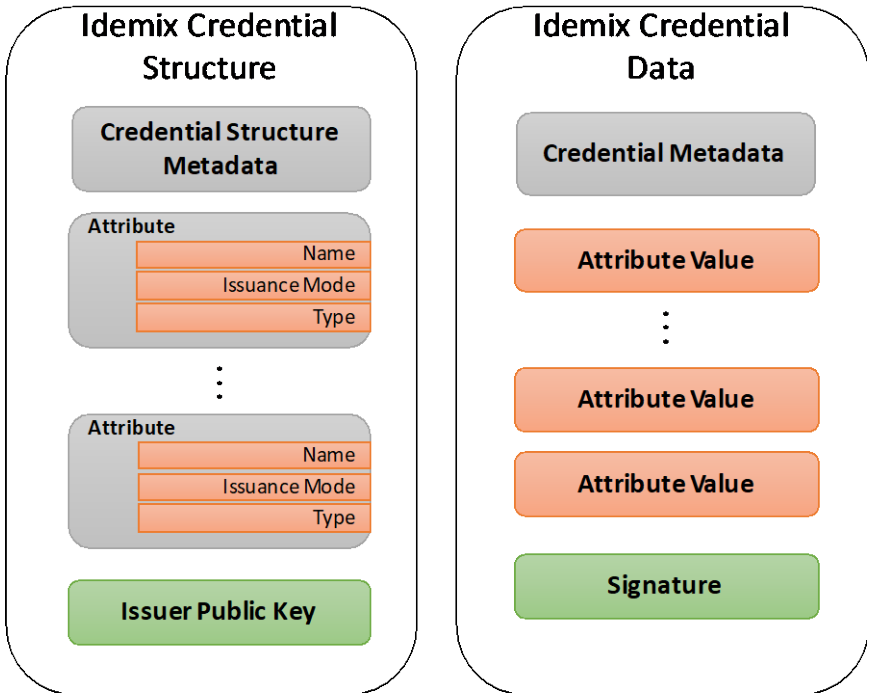


Figure 21 Mapping of Online Service and User Device respectively to a Verifier and a Prover running the Proving Protocol to allow the proof of possession of credentials

3.4.1.2 Idemix Credential Format

The most popular Idemix implementations make use of a credential format that is based on XML. The credential structure is kept separated from the credential data (i.e. attributes specification). In this way Idemix provers and verifiers can transfer credential structural metadata, without disclosing any other information about the actual credential contents. Verifiers need the credential structure information when receiving a proof, in order to extract its semantics. Idemix credential structures are published by the issuers. Actual credentials contain attribute data, follow the format specified in

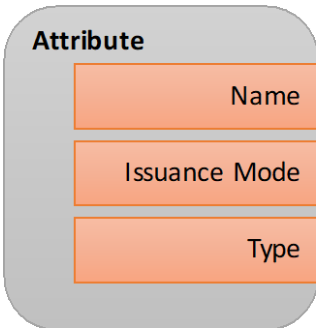
the published credential structures, contain the recipient’s master secret key, and are issued to a recipient’s pseudonym, through the issuing protocol.



Although the SAML [26], encoded in XML like the Idemix credentials, is usually associated to the Single-Sign-On (SSO) [18] profile, its flexibility allows it to be used in other contexts. Idemix credentials could thus be specified using SAML: this would allow for a layered and extensible approach, in which SAML acts as a base transport format for Idemix (and other) credentials (and proofs) employed throughout ReCRED.

3.4.1.3 Idemix Attribute Format

Each Idemix attribute definition in an Idemix credential structure definition, is composed by a name, an issuance mode and a type. The name is a unique (within the credential scope) label for the attribute. The issuance mode distinguishes between attributes which are known to the issuer, attributes for which the issuer has a commitment, and attributes which are hidden from the issuer. The type denotes the attribute data type: string, int, date or enum.



The attribute value in a credential is encoded according to its type. A prover can choose which attributes to reveal when creating a proof of possession of credentials. Thus we can distinguish between revealed attributes and unrevealed attributes.

3.4.1.4 Idemix Protocol

The Idemix protocol requires that all parties agree on public master system parameters such as the bit length of all relevant parameters as well as the groups to be used. Given such global parameters, a prover can choose her master secret key that will be contained by each credential, resulting in a parameter that binds together all credentials. This prevents from sharing credentials with the aim of collusion, as sharing one credential effectively implies sharing all the credentials of a user. Moreover, the master secret allows the prover to derive the pseudonyms to use when she requires credentials to be issued by the issuers. Each receiver can generate different pseudonyms to be shown to the issuers. Such pseudonyms, even if generated by the same receiver, cannot be linked to each other unless she proves that they are based on the same master secret key. Issuers generate public and secret keys associated to the cryptographic primitives they use and make the public keys available together with a specification of the services they offer. As an example, each issuer publishes the definition (i.e., the Credential Structure as shown in Section 3.4.1.2) of the credential it allows to be issued. To obtain a credential, the receiver contacts an issuer and agrees with her on the structure of the credential, i.e. which will be the values of the attributes asserted by the credential. She then runs the interactive issuing protocol with the organization. Having acquired a credential, the receiver switches to the role of a prover in order to prove to a verifier the possession of the credential. A proof of possession may involve several credentials acquired by the same user or proving statements on the attribute values contained in the credentials using the proving protocol. Moreover, these proofs may be linked to a pseudonym chosen by the prover. Moreover, the proving protocol and issuing protocol credentials may be combined, in the case of an issuer requiring the recipient to release certified attribute values (a proof that she holds a credential issued by another party) before issuing a new credential. The Idemix protocol consists of three basic functionalities described in the following sections: i) system setup, allows parties to get initialized in the Idemix system, ii) credential issuance, is the functionality that permit a receiver to get the credential by the issuer and finally iii) credential proving, is the functionality demanded to the verification of credentials presented by a prover to a verifier.

3.4.1.5 System setup

The anonymous credential system requires general parameters, which we separate into system parameters consisting of bit lengths, and group parameters, which define the groups that are used within the underlying cryptographic scheme. In addition, the issuer and receiver/issuer must generate parameters to be able to participate in the credential system. The issuer’s key pair is used for issuing certificates, that is, issuing signatures on lists of attributes. The maximum number of attributes of the credential is determined by the related public key length and the number of reserved attributes (e.g., the master secret) that are not allowed to be issued. The user chooses a uniformly random master key and is able to generate as many pseudonyms as she wants. Each pseudonym is un-linkable to any other pseudonym generated by the user. However, it is possible to

define also domain pseudonyms in which case a user can generate only one pseudonym per domain (i.e., given the domain and the user’s master secret key, the domain pseudonym is unique).

3.4.1.6 Credential Issuance

The Credential Issuance protocol is an interactive protocol in which the recipient of a credential (e.g. the User’s Device) and the issuer of the credential (e.g. the Identity Provider) participate. When this protocol runs, the issuer and the recipient interact to agree on a signature that will be the cryptographic component of the credential owned by the recipient at the end of the handshake. At each step of this three-way protocol, a zero-knowledge proof ensures that both parties are correctly implementing the protocol. It is possible to update a credential by running again at the issuer side some parts of the issuance protocol.

3.4.1.7 Credential Proving

The prover can use a proof specification to create a proof of possession of a credential. This can be verified by a verifier. The credential proving protocol does not need to be as much interactive as the issuance protocol. At first, the prover builds a non-interactive proof of the statements defined in the proof specification, i.e. the proof of the possession of credentials required by the verifier. Many assertions could require multiple credentials from different issuers. For example, a service that is verifying if a user matches its policy, might want to ensure that the age field of the government ID credential matches with the ownership of a driver license credential indicating that the prover is 18+.

3.4.1.8 IRMA

IRMA (I Reveal My Attributes) project is the starting point for the implementation of the reference architecture for the ABAC infrastructure of the ReCRED framework. IRMA technology is open-source and provides a basic infrastructure on top of which new commercial services can be developed. Such infrastructure perfectly fits also the ReCRED requirements providing a fully-decentralized infrastructure in which the user is the only owner of its own credentials (it does not rely on third-party identity brokers). The IRMA architecture in Figure 21 is composed of the following components:

- **IRMA API Server**: this is a server that sits between the user device and service or identity providers on the other hand. It handles all specific cryptographic details of issuing credentials and verifying disclosure proofs on behalf of the service or identity provider.
- **Application Server:** this is the service or identity provider server implementing the interface between the user and the IRMA API Server (cryptographic API)**.**
- **User Device:** this is the user device used to access services using IRMA application and enforcing ABAC by means of Idemix credentials.

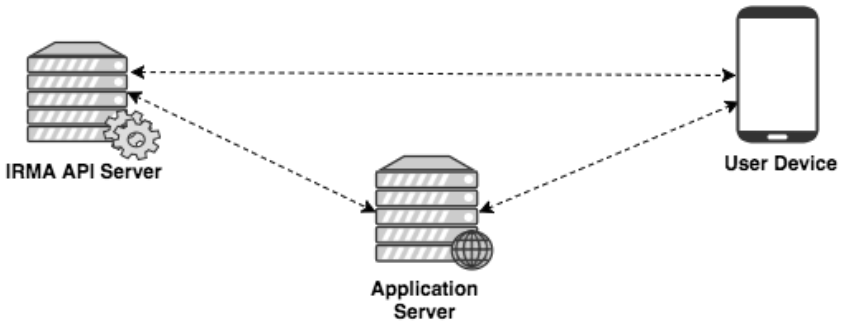


Figure 22 IRMA Architecture

The design of the IRMA architecture allows to have cryptographic implementation clearly separated with respect to the application level that is service/identity-provider specific. In what follows a brief description of the verification and issuance phase of the IRMA credentials are described.

3.4.1.8.1 IRMA Verification

The IRMA protocol for the verification phase of the credential is quite simple and allows to be adapted to the ReCRED use cases. The verification protocol is followed in the case of a service provider that requires to verify the credentials of a user that asks to access a service.

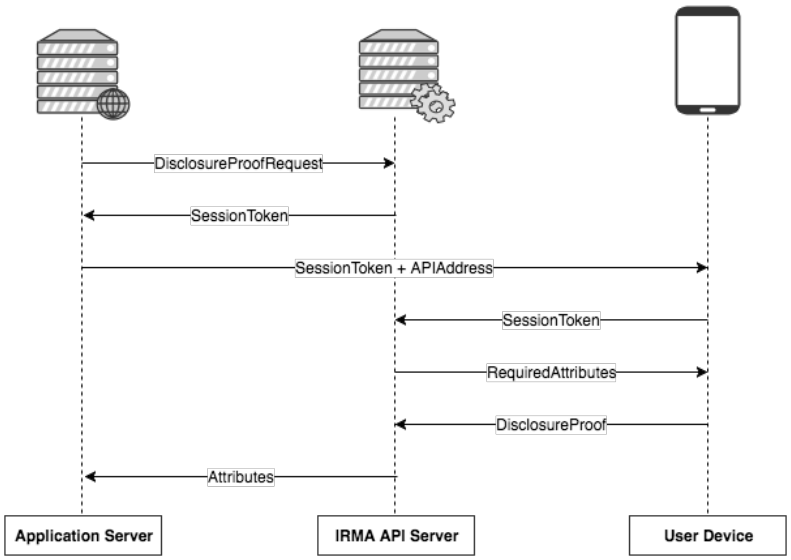


Figure 23 IRMA verification protocol

The verification protocol’s phases reported in Figure 22 are the followings:

1. The Application Server submit the **proof request** to the IRMA API Server
2. The IRMA API Server provides to the Application Server a **session token** to be provided to the User Device together with the end-point that the user should contact to provide the **proof**

3. The User Device access to the end-point provided by the Application Server by using the **session token** to receive the **proof request**
4. The User Device provides the **proof** required by the IRMA API Server on behalf of the Application Server. The IRMA API Server verifies it and sends the results of the verification result to the Application Server.

At the end of the protocol the Application Server allows the User Device to access the service based on the verification steps results. Note that the Application Server is not required to run any cryptographic operations.

3.4.1.8.2 IRMA Issuance

The workflow here will be much the same as with disclosing. The issuing protocol is followed in the case of a user requiring the issuance of credentials from an identity provider.

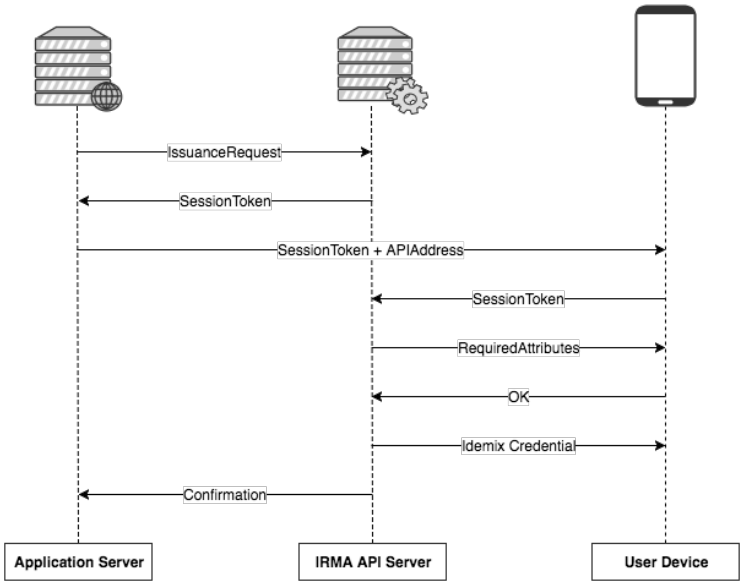


Figure 24 IRMA Issuance Protocol

The issuance protocol’s phases reported in Figure 23 are the followings:

1. The Application Server submit the **issuance request** (triggered by the user) to the IRMA API Server
2. The IRMA API Server provides to the Application Server a **session token** to be provided to the User Device together with the end-point that the user should contact to require the **credential**
3. The User Device access to the end-point provided by the Application Server by using the **session token**
4. The User Device is issued of the Idemix **credential** by the IRMA API Server

At the end of the protocol the User Device stores the received credential on a secure storage in the device. Note that, even in this case, the Application Server is not required to run any cryptographic operations.

3.4.2 U-Prove

The technology behind U-Prove [30] is based on cryptographic primitives including public-key cryptography on subgroups or elliptic curve constructions, and secure hash algorithms.

At the core of the U-Prove technology is *the U-Prove token*. A token is comprised of a data collection (e.g. attributes), and is cryptographically protected against tampering. Tokens are issued by an authoritative source (*Issuer)* to a user (*Prover*) through an *issuance protocol*, and subsequently presented by the user to a Relying Party (*Verifier*) through a *presentation protocol*.

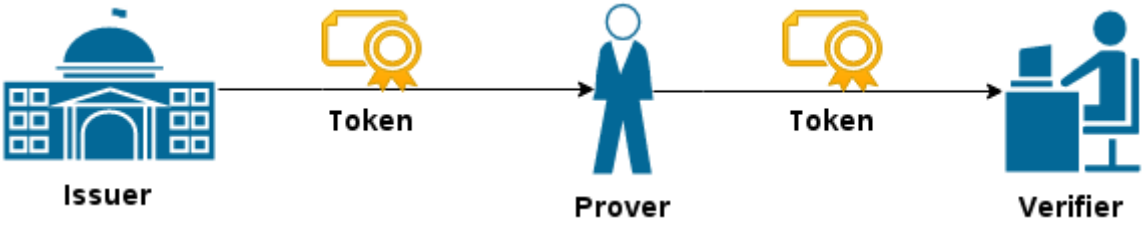


Figure 25. U-Prove Roles

As it can be seen from Figure 25 the main three roles in U-Prove are Issuer, Prover and Verifier. Please note that the mapping of the U-Prove roles to the ReCRED architecture entities are the same as described in the Idemix Section. In this section, we shall provide some U-Prove specific details with the U-Prove specific roles.

A universally *unique token identifier* can be computed from each U-Prove token but the Issuer cannot learn any information about this value at issuance time; hence it cannot be used to correlate a presented token to its issuance instance. Token identifiers can be used to identify returning visitors and for token revocation.

The verification of a U-Prove token and a corresponding presentation proof requires only an authentic copy of the Issuer parameters under which the token was issued. The issuer parameters are generated and distributed by the Issuer.

The Prover can present a pseudonym computed from one token attribute which is unique to a particular scope defined by the Verifier. This allows the Verifier to recognize the Prover on repeated visits even if a different token is used, as long as it encodes the same attribute used in the computation of the pseudonym.

3.4.2.1 Issuer parameters

Issuer parameters have the following components

$$UID_p, desc(G_q), UID_H, (g_0, g_1, \cdots g_n, g_t), (e_1, \cdots, e_n), S$$

where:

- UID_p is an application-specific unique identifier for these parameters;
- $desc(G_q)$ specifies a group $G_q$of prime order q;
- UID_H is an identifier of a cryptographically secure hash algorithm;
- $(g_0, g_1, \cdots g_n, g_t)$ is the Issuer's public key;
- $(e_1, \cdots, e_n)$ is a list of boolean values indicating whether or not the attribute values $(A_1, \cdots, A_n)$ are hashed when computing a token's public key *h*.
- S is a string which holds an application-specific specification of these parameters and U-Prove tokens issued using them.

The value *n* is application specific and represents the number of attributes encoded into *each* U-Prove token that will be issued using these Issuer parameters.

Practically, the Issuer parameters are equivalent to the Certificate Authority's certificate in a PKI environment.

3.4.2.2 Token content

A token has the following form

$$UID_p, h, TI, PI, \sigma_z', \sigma_c', \sigma_r', d$$

where:

- UID_p is an application-specific unique identifier for the Issuer parameters under which this token was issued;
- $h$is the public key of the token;
- TI or *Token Information* field is used to encode token-specific information which is always disclosed to Verifiers, such as token usage restrictions, a validity period, or any other metadata;
- PI or *Prover Information* field. This field is used to encode Prover-supplied information, hidden from the Issuer;
- $\sigma_z', \sigma_c', \sigma_r'$ form the Issuer's signature on token fields;
- d is a Boolean flag which indicates if the token is protected by a secure device.

The attribute fields contain values which are encoded by the Issuer, and must be known by the Prover. Attributes can take any values as long as their unsigned integer representation is less than the prime order of the group construction.

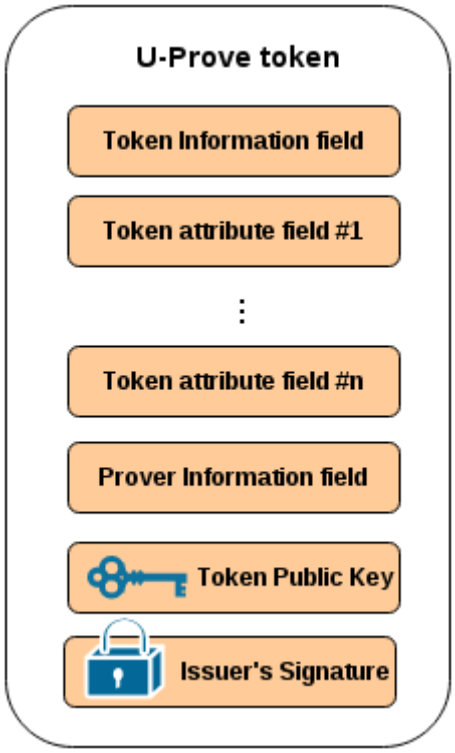


Figure 26. U-Prove token public components.

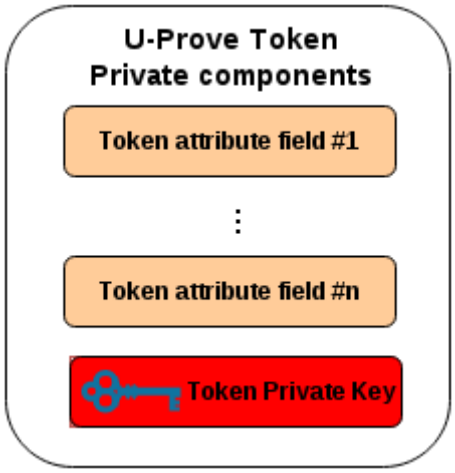


Figure 27. U-Prove token private components.

The private key corresponding to the U-Prove token, is generated by the Prover during the issuance protocol.

Tokens are issued by the Issuer, are cryptographically protected against manipulation and exhibit the following properties:

- Integrity and source authenticity: A secure digital signature of the Issuer over the entire contents of the token is created by applying its private key. This signature can be verified by

any third party in possession of the Issuer’s parameters, and authenticates the fact that this token was issued by the Issuer and is genuine.

- Replay attack prevention: Every U-Prove token contains a specific public key that is known only to the Prover since the Prover randomly generates it during the issuance protocol, together with a corresponding private key for the U-Prove token. The private key is not part of the token, and is never disclosed by the Prover.

3.4.2.3 Token issuance

The Prover obtains a U-Prove token from an Issuer by participating in an instance of the U-Prove Issuance Protocol. The cryptographic protocol requires as input, among others, the attributes which are going to be encoded into the token. Assuming both parties know which attributes are going to be encoded in the token, the protocol requires the exchange of three messages (Issuer to Prover, Prover to Issuer and lastly, Issuer to Prover).

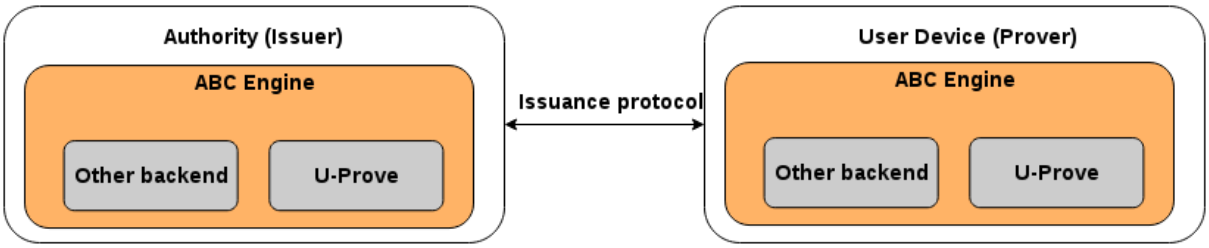


Figure 28. Parties and components involved in the Issuance Protocol.

Since the Issuance Protocol does not include the negotiation of the attributes and does not check the Prover’s eligibility to receive a token, the protocol needs to be augmented with other messages.

An Issuer can issue arbitrarily many U-Prove tokens by using the same private key, without any loss of security. The public key is part of the Issuer parameters and is available to any third party interested in verifying an issued U-Prove token.

3.4.2.4 Token presentation

Issued tokens are presented to the Verifier by transmitting a copy of the token, the attribute values which are going to be disclosed, and a Proof of Possession (PoP) generated by making use of the private key corresponding to the token over a message and the non-disclosed attribute values.

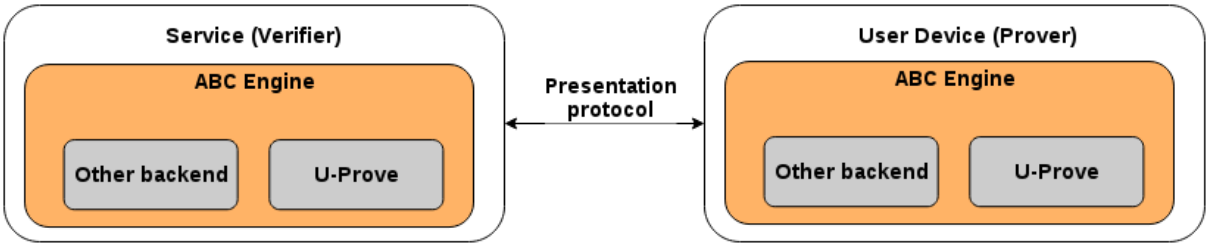
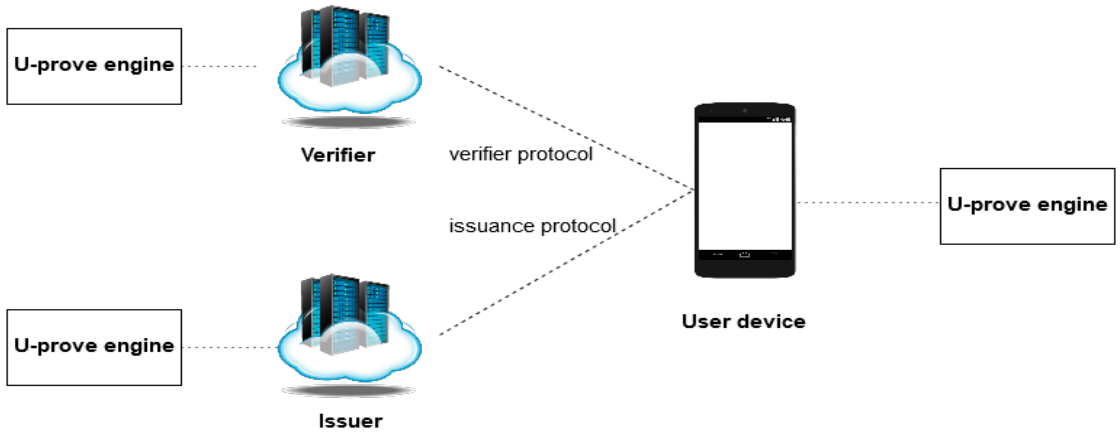


Figure 29. Parties and components involved in the Presentation Protocol.

Although the base specification of U-Prove does not specify functionality like revocation, identity escrow or predicate proofs on attributes, these can be implemented, since the Prover can generate cryptographic commitments to any attributes, which allows an external module to provide additional functionality. Specifications for extended functionalities are already available [9][7][8][24][23][22][15][21][6].

3.4.2.5 U-prove engine

In order to be able to run the U-Prove protocol on an Android device we used the official C# SDK [33] provided by Microsoft, which we adapted on Java programming language. To create a fully-decentralized infrastructure we need to add the engine on the user device as well as on the issuer’s and verifier’s server. The issuer will act as a REST server and will expose specific functionalities for generating U-Prove tokens and for setting the issuer parameters. The verifier will expose only one functionality to verify if the prover owns a series of attributes . In the following image is depicted the general U-Prove architecture.



3.4.2.6 Attributes provider

Given that the U-Prove protocol does not offer a way to transmit the attributes to the issuer we use the Identity Management Module from the Identity Consolidator to deliver trustful user attributes. When a request for generating a U-Prove token is received by the Identity Consolidator, it verifies against a database if the user attributes are valid. If the Identity Management module returns positive response, then the attributes are sent to the Credential Management module, which will generate the U-Prove token. Additional information may be sent by the prover along with the attributes, such as token information or the fact that the token generation will be protected by a device. This additional information depends on the token type generated by the prover (-U-Prove or Idemix). The Credential Management module will have a FIWARE interface, which will permit U-prove and Idemix abstractization.

The following image presents the token issuance process.

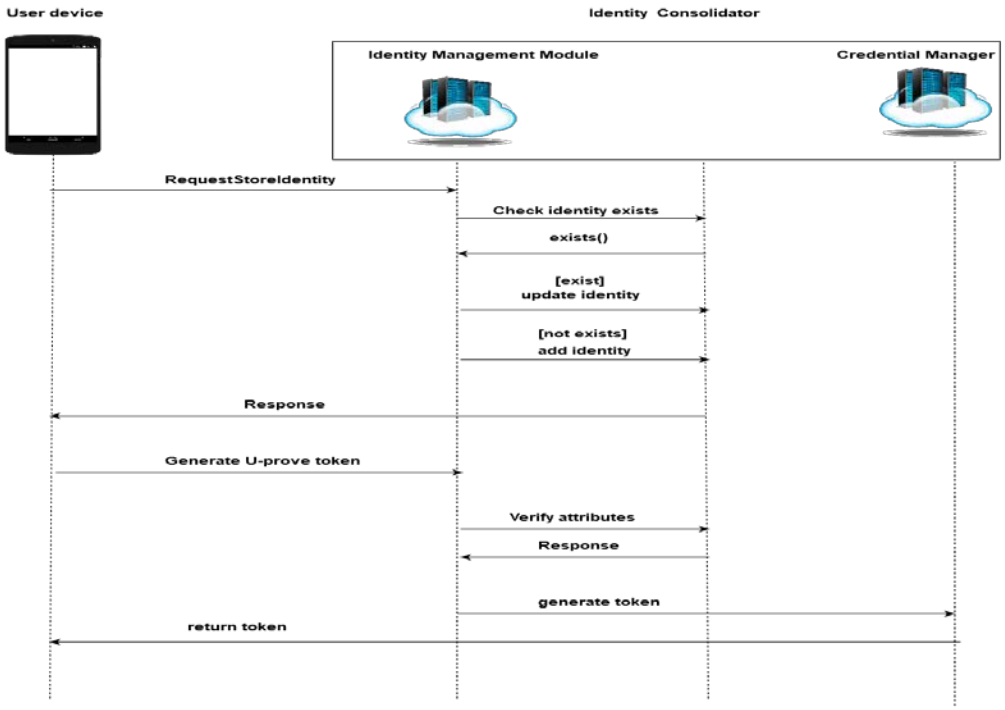


Figure 30 – U-Prove token issuance

3.4.2.7 Client-Server architecture

In order to have a modular design and to ease the module integration we used a client-server architecture. In order to permit the U-Prove artifacts transport, a serialization module was designed. The serialization process acts like an extension for the U-Prove engine and it is responsible for transforming U-Prove objects into JSON messages. Both the issuer and verifier follow this client-server paradigm. The next component diagram presents the process of sending data between the client and the U-Prove server.

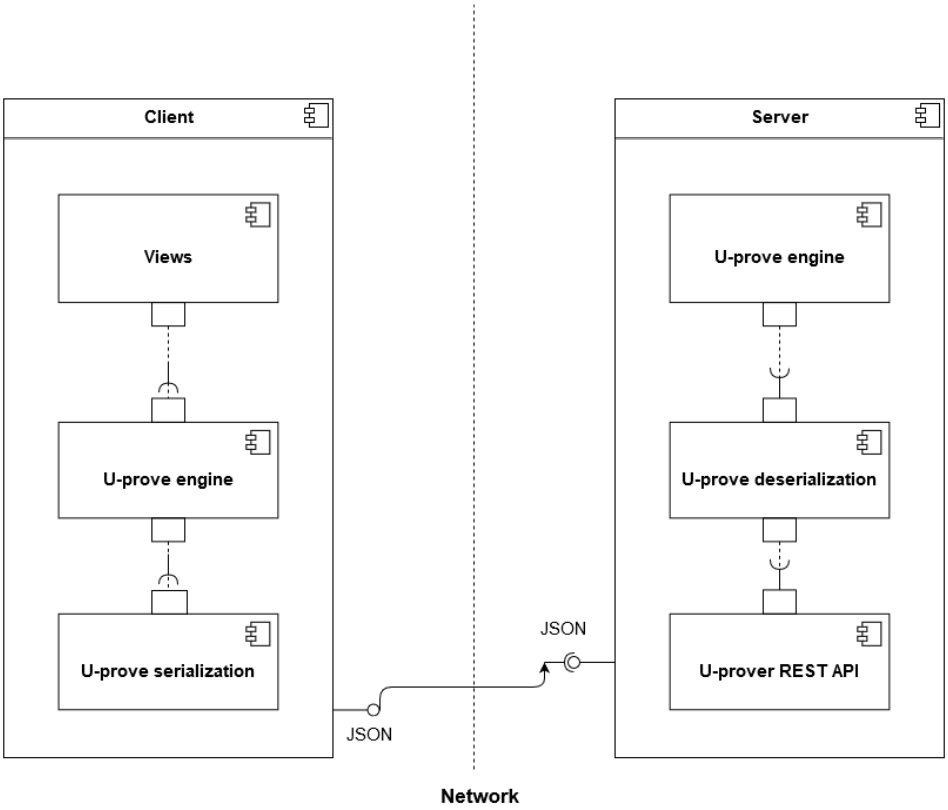


Figure 31 – U-Prove client-server architecture

3.5 ReCRED integration

Idemix, as well as U-Prove, are the building blocks of the ReCRED platform attribute-based access control. Idemix will enable the ReCRED framework to support anonymous, un-linkable and un-traceable credential issuing and proving. Existing Idemix implementations [17] and [16] for the mobile implementation will be integrated in most of the components of the ReCRED platform as described below:

Identity Provider: acting as an issuer is able to maintain a history of the released credentials, as well as the context of the attributes it can issue. By running the issuance protocol, it is able to directly issue idemix’s credentials to the recipient party (e.g. the user’s device).

- **User Device**: is the central component of the ReCRED device-centric authentication. It acts both as a recipient requesting credentials to issuers and as a prover. Note that, since it is a mobile device, the integration of Idemix primitives will be mainly based on [16]. IRMA’s features match the requirements for Idemix components integration in user devices: it is already ported to a mobile platform and the formats employed for attribute exchange are based on languages that are supported by other ReCRED components. However, such

integration in the ReCRED user device will require to enforcing the storage of credentials on a TPM that is not considered in the IRMA implementation.

- **Identity providers and Identity Consolidator**: could act both as an issuer and a verifier. The Identity Consolidator or IdP is demanded to issue credentials that are reflected by identity-attributes directly to the user device. Moreover, the identity consolidator may require the user device to prove the possession of credentials, for example, in case the user should demonstrate the ownership of her account (e.g. password recovery procedure). After the verification of the credentials, the Identity Providers or the Identity Consolidator will release the verified identity attributes to the Service Provider. Subsequently, the SP will be able to check the attributes against his policy to determine whether access should be given to the user.

3.5.1 FIWARE interfaces

The ReCRED project aims to support an Attribute Based Access Control Infrastructure that allows to use anonymous credentials and it is based on technology at the state of the art as Idemix and Uprove. Such technologies are the core functionalities of the PABAC architecture (i.e. providing just the crypto functionality) and have different requirements for the interfaces. FiWARE project provides a common interface between Idemix and Uprove by means of the Privacy Generic Enabler. In the ReCRED ABAC architecture we aim to pair together the outputs both from IRMA and FiWARE projects to provide an integrated ABAC architecture, shown in Figure 29, providing both Idemix and Uprove credentials support.

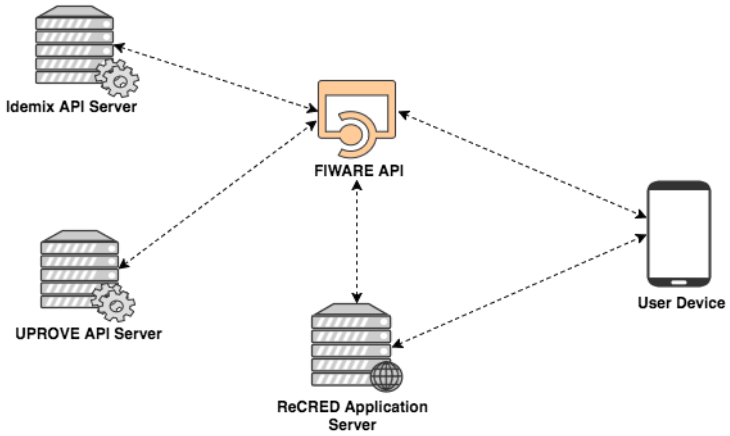


Figure 32 ReCRED PABAC architecture with FIWARE APIs

As in IRMA Architecture, we aim to clearly separate the cryptographic logic (API Server) from the application logic (Application Server). Moreover, the ReCRED ABAC infrastructure will use the FiWARE interface for the issuance and verification of credentials between the User Device and the API Server.

3.6 gateSAFE

Organizations usually have multiple web applications for different purposes. Applications come in many flavours, from different vendors, are independent of each other and have different authentication mechanisms. Individuals are asked to have users and passwords for each application, and have them changed according to the Security Policy.

Given this situation, Identity management becomes a real problem. A solution to this issue would be to harmonize the authentication of all applications and have a single identity database by changing the applications. Most likely this is not possible given the variety of the programming languages being used and most applications are closed source.

gateSAFE enables secure access, accounting and control to both modern and legacy web applications by leveraging state of the art technologies like Transport Layer Security and Digital Certificates.

Transport Layer Security is a suite of cryptographic protocols and signalling that provide communications security across a network in a way designed to prevent eavesdropping and tampering. As the central point of access for web applications, gateSAFE acts as a gateway, securing the communication with the client and accessing the requested resource on client's behalf.

By using Digital Certificates and doing the authentication behind the scenes, gateSAFE solves the Identity management and fragmentation issues in an organization.

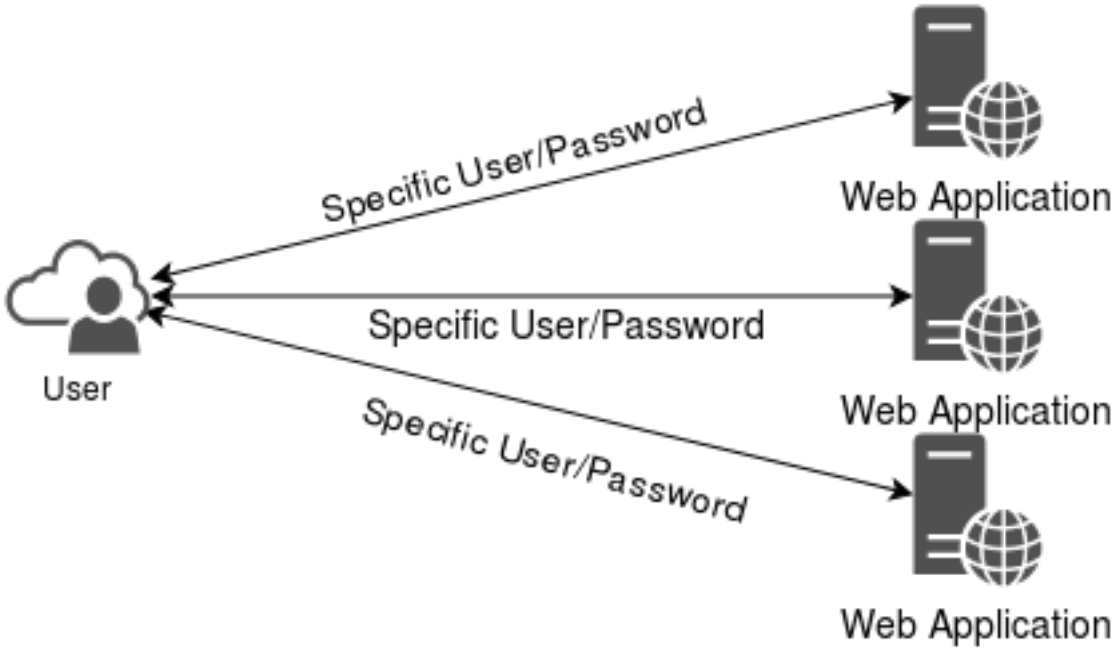


Figure 33 Usual architecture where users have a user and password for each web application

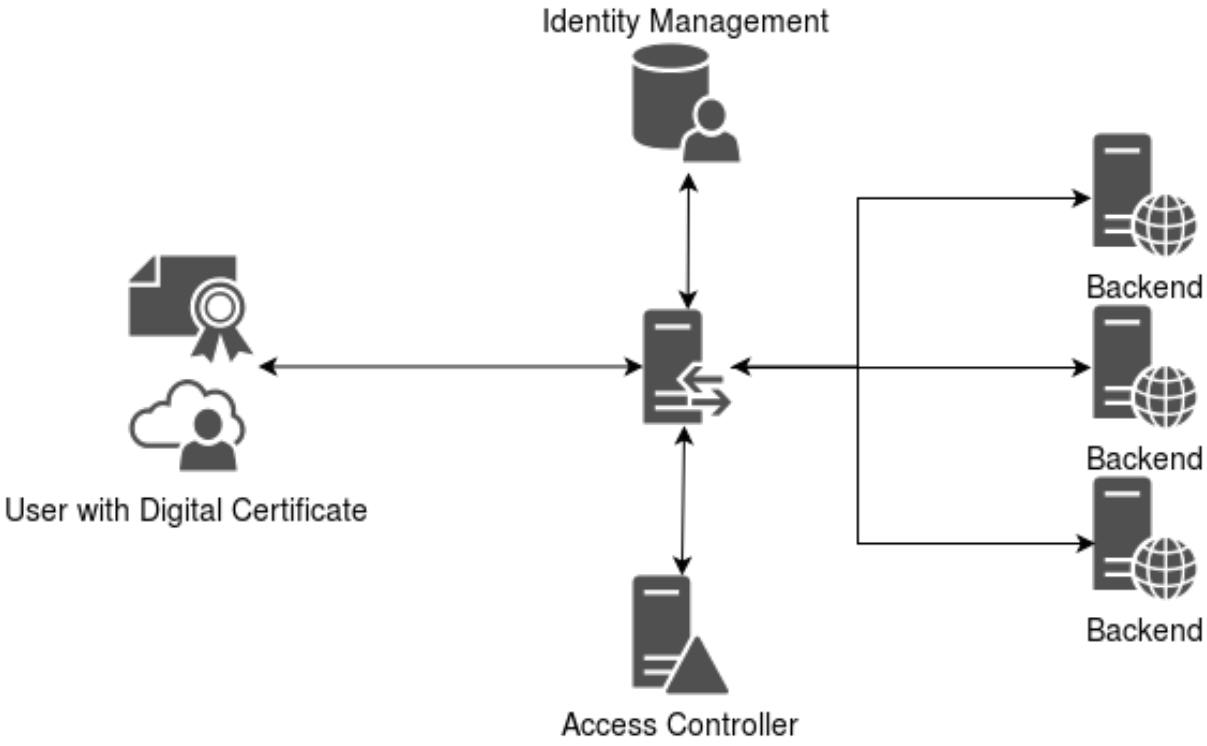


Figure 34 gateSAFE makes it easier to manage user identities by using Digital Certificates

3.6.1.1 Authentication

gateSAFE is compliant and follows the Public Key Infrastructure (PKI) industry standards.

Mutual client authentication (i.e. the client verifies the server and the server verifies the client) is based on TLS using X.509 Digital Certificates.

Hardware-backed keys stored on smartcards offer higher security levels, and require user interaction to supply the PIN protecting the access to the keys.

Certificate validation is done using the organization's existing Public Key Infrastructure (PKI) services such as Certificate Revocation Lists (CRLs) provided offline or downloaded through LDAP and Online Certificate Status Protocol (OCSP).

3.6.1.2 Backends

Backends are represented by any Web Application accessible through a HTTP/HTTPS/WebSocket URL. When clients connect to gateSAFE URL, such as https://gatesafe.certsign.ro/{wiki,email}, the application does a backend mapping to the specific web application.

In a scenario where /wiki maps to http://<internal-ip-1>/wikimedia, the gateway will open an HTTP connection to this backend, forward the client's request, parse the reply, and forward it back to the client.

This forwarding is transparent to the client since a rewrite engine takes care of replacing the Uniform Resource Identifier (URI) addresses with the correct ones.

3.6.1.3 Single Sign On

X.509 Digital Certificates contain fields with information related to the certificate holder.

These fields, along with other information from gateSAFE's database is used to authenticate the users to the Web Application running on a specific backend.

Consider a legacy application requiring an HTTP Basic Authentication. To have a unified authentication mechanism, gateSAFE can do the authentication on the client's behalf. Currently the following authentication mechanism are supported:

- Basic Authentication
- Form-based Authentication
- LDAP Authentication

Other authentication mechanisms are supported through use of plugins which can be developed by third parties using the provided Software Development Kit (SDK).

3.6.1.4 Access Control

Every backend and specific URIs must have a security level specified in configuration files.

Users are associated to a security level by having their fields from the Digital Certificated matched in a Security Level Filter.

On every user request to access a specific Web Application, the Access Control module is used to decide whether the user's request is allowed or rejected.

3.6.1.5 SSL Termination

The simplest approach for an application to enable communications of different types is to use web protocols to establish communication channels, and inside those channels to exchange application specific messages (e.g. almost all modern chat applications moved towards HTTP protocol to exchange messages). To support that, HTTP 1.1 keeps connections alive and introduces compression for faster transfer times. As that is not enough, HTTP 2.0 and HTTP WebSockets offers a more reliable connection to support live communications and more fragmented resources.

As more and more communication moves into web, security of the communication became mandatory and having specialized components that handles communication encryption moved from a recommendation to a must. As a reverse proxy, gateSAFE terminates the TLS protocol and handles all the TLS aspects, such that the application that is proxy-ed does not need to know about different security aspects that TLS offers (e.g. support protocols like OCSP/LDAP or security aspects like SSL attacks and hardware keys storage); therefore, gateSAFE offloads the application such that the application handles the specific business logic only.

3.6.1.6 Logging modules

A logging mechanism is a very important feature for any authentication application as it provides, in the first place, accounting. A rich set of options to control logging messages are available. Log data can be written to the following sinks:

- Files;
- Digitally Signed file providing non-repudiation, integrity and tampering protection;
- SQL Server database;
- SQLite database.

In gateSAFE, the logging mechanism is implemented in all high-level components and can enabled or disabled by the simple mean of configuration, such that different logging levels can be obtained: from the error level, which provides the least information and up to debug level which is the most verbose logging level. In parallel, accounting level provides information about every web resource that a specific user accessed over time.

Filters are a native feature of the logging subsystem. With filters, you can control which fields get logged, and under which conditions. Possible fields which can be logged:

- Any field from the client's digital certificate
- Time and date
- Accessed URI
- HTTP Traffic
    - Requests
    - Responses

The native C++ SDK allows 3rd party users to implement their own logging and filtering mechanisms.

3.6.1.7 Accounting

Accounting is an important part of the logging modules and it allows accounting on a fine-grained level since every user is uniquely identified by the Digital Certificate's Serial Number.

Possible sources of accounting:

- Number of visits (requests, responses) of each user
- Number of visits on every Web Application
- Generated traffic

3.6.1.8 Real time traffic monitor

In gateSAFE, the application administrator or another person has the possibility to watch, in a live environment, all the accesses that the users do to the protected resource. The traffic monitor is a specific logging module as well, derived from the logging engine.

3.6.1.9 Security

Security is an integral part of the overall product. gateSAFE supports using Hardware Security Modules (HSM) for Public Key cryptography operations involved in the Transport Layer Security protocol.

For deployments, which have lower security requirements, software keys stored on the server's storage give the same level of security as the overall Operating System.

HSMs have several important benefits over software keys, the most important one being the fact that the private key cannot leave the module in case of a security breach.

3.6.1.10 Scalability

Scalability is an important aspect of any business which must be prepared for an increasing number of clients and users.

gateSAFE is a multi-threaded application and makes efficient use of multi core and multi-processor server hardware.

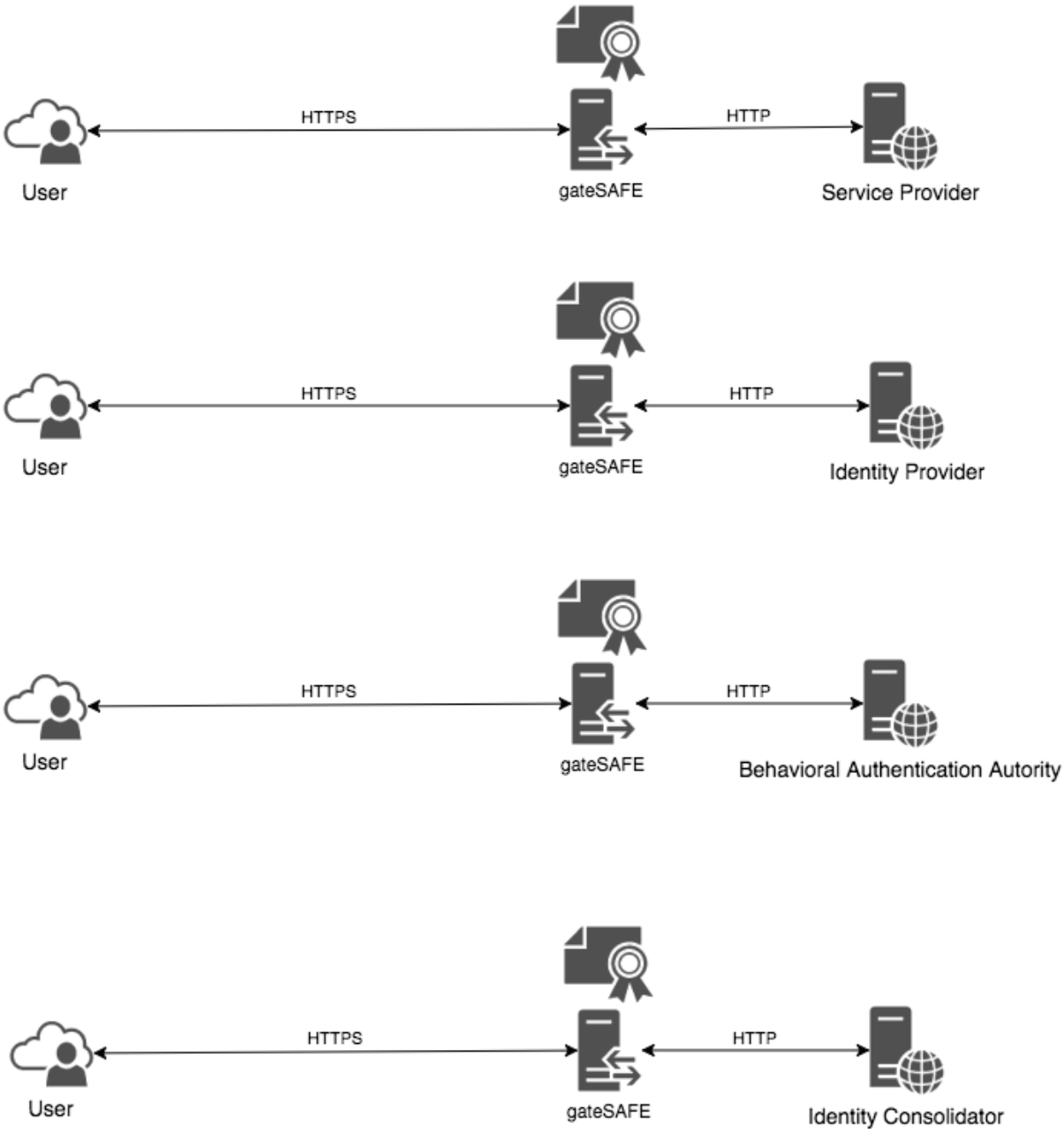
Every backend can have multiple URLs defined, allowing gateSAFE to acts as a load balancer for every Web Applications.

gateSAFE supports working in a multiple instances deployment, where load balancers are used to redirect users to different instances.

3.6.2 gateSAFE in ReCRED

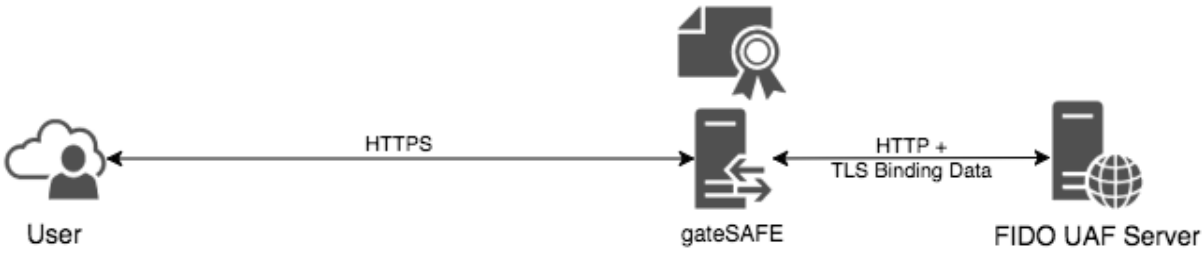
As depicted in the Figure1, in legacy implementations, a client has one password for each web application, corresponding to one account. Before ReCRED, gateSAFE solved the passwords problem by introducing X.509 digital certificates authentication for the clients. Therefore, clients did not use passwords anymore to authenticate to the web applications, but used one single digital certificate to authenticate to gateSAFE and further, it’s gateSAFE that handles the authentication. Although X.509 become a standard of authentication in enterprise environments, because user identity is already known in that environment, there are situations, like ReCRED, where knowing the user identity is not desired to perform authentication and authorization.

Introducing gateSAFE in ReCRED, allows us to take advantage of the gateSAFE features like TLS endpoint and to extend, further, gateSAFE to support the new authentication protocols that are more privacy preserving for the end-user, like FIDO UAF, like OAuth and U-Prove.



gateSAFE Integration with ReCRED services

For an improved security, there are some authentication resources for which specific TLS information need to be shared between the proxy and the web application. Acting as a reverse proxy, gateSAFE add specific TLS connection data in the HTTP protocol. For FIDO UAF protocols (e.g. registration, authentication, deregistration), there is a specific binding that FIDO UAF Server must implement by adding information from the underlying TLS protocol into FIDO UAF messages. This information binding must be realized together with gateSAFE as the latter must be aware of the protocol used.



gateSAFE Integration with FIDO UAF Server

As gateSAFE is the TLS endpoint, to make possible binding of TLS channel, it must share with FIDO UAF Server information from TLS protocol, as specified in RFC5929, one of the following:

- TLS channel ID
- Server End-Point
- TLS Server Certificate
- TLS Unique

This is realized through the HTTP by adding, in the HTTP Headers, the information needed by FIDO UAF Server on the specified resources.

gateSAFE is used to support the TLS implementation for the following entities: Service Providers, Identity Providers, Behavioral Authentication Authorities, Identity Consolidator and for the services that indirectly implements them: OpenID Connect, FIDO UAF Server, Credential Management Module.

gateSAFE is not forcibly bound to the entire architecture; in simpler environments, gateSAFE can be used to implement only a specific authentication protocol, together with one only authentication component (e.g. FIDUO UAF Server or OAuth) that can be regarded as a gateSAFE authentication module.

3.7 XACML

3.7.1 Introduction

XACML is a standard that is used for describing a policy language and an access control decision request/response language that is based on XML. The policy language describes general access control requirements that allow the definition of new data types, combining logic etc. The request/response language enables you to query and ask whether or not a given action should be allowed (Permit/Deny). In summary we use XACML because it has the following advantages:

- **Standard**: Much easier to be reviewed and adopted by the community because it is a standard language. Also, it can interoperate with other applications without significant issues.
- **Generic**: It can be used in any environment. For example, a policy can be written for a particular environment and then be used by other applications in different environments.
- **Distributed**: A policy can be written that refers to other policies in arbitrary locations. This allows different people to manage sub-pieces of policies and XACML is able to combine the results and produce a single decision.
- **Powerful**: The XACML language supports a wide variety of data types, functions and rules. This enables developers to accurately define complex access control policies.

3.7.2 XACML Architecture

In summary the XACML architecture consists of five components that are shown in Figure 35.

- **Policy Enforcement Point (PEP):** Component that is responsible for intercepting user’s requests for accessing a resource and makes a request to the PDP in order to obtain a decision (Permit/Deny) and acts accordingly.
- **Policy Decision Point (PDP):** This is the component that performs the evaluation of access requests against the defined access control policies. The PDP produces the decisions of the access requests.
- **Policy Administration Point (PAP):** The PAP is used for managing the access control policies. This includes the addition of new policies and the edit/deletion of existing ones.
- **Policy Information Point (PIP):** The PIP component is an entity that is used to gain access to important information that is used for access control. This includes the types of available resources, environment details, etc.
- **Policy Retrieval Point (PRP)**: The PRP is a logical entity that is used to represent the entity that is responsible for storing the XACML policies. Typically, a database is used for implementing a PRP.

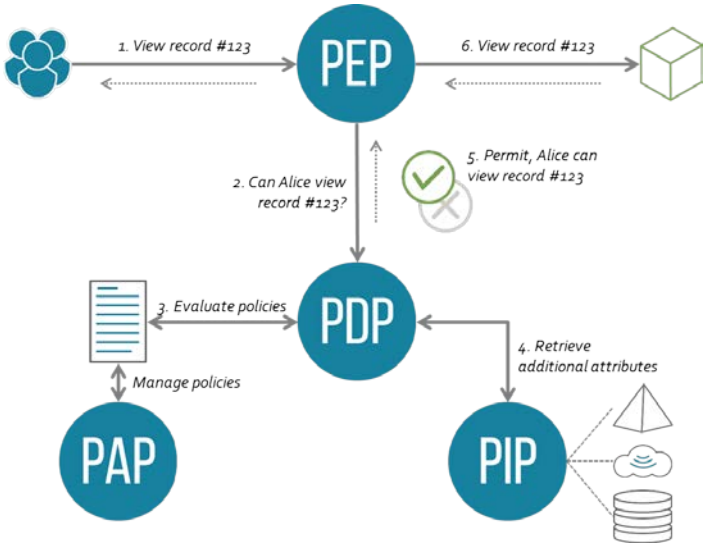


Figure 35 XACML Architecture (Source: Wikipedia)

3.7.3 XACML and ReCRED

XACML is the basic building block for implementing the Access Control Reasoning tool on the Service Providers entity within the ReCRED architecture. Within the ReCRED project, we will implement the abovementioned components of XACML. Specifically, for the PIP and PRP components we will make use of a database that will be responsible for storing the access control policies and different environment-specific details. For the PAP we will implement a Web Application where the Service Provider administrator will be able to add/view/edit/delete access control policies. Upon completion of these actions, the changes will also be applied to the PRP component. For the PEP and PDP we will implement dedicated back-end components that will be responsible to intercept identity attributes that will be retrieved from Identity Providers using the OpenID Connect specification, form a new XACML request and decide using the defined access control policies.

To conclude the PAP component, will be enhanced with Machine Learning and Data Mining algorithms so that the Service Provider can get recommendations for editing/creating/deleting access control policies. This will enable Service Providers to offer a much better solution for providing access control to end-users.

4 Conclusions

The ReCRED architecture presents the components of the system and the protocols and interfaces used to communicate between them.

These are further detailed in the deliverables from the work packages 3, 4 and 5:

- **WP3 – Beyond the Password: Device-centric Authentication** describes DCA protocols, user/device and device/server interfaces and APIs, including description of needed extensions to existing standards with multi-level authentication and recovery mechanisms. Moreover, it documents the access based on behavioral and physiological biometrics and how the trusted device execution environment can be exploited for human-to-device authentication.
- **WP4 – Identity de-fragmentation and real-world binding** deals with the identity management issues. It matches the multiple virtual identities of users, unifying (de-fragmenting) them under a single profile. In addition, it completes the profile with the physical attributes of the user.
- **WP5 – Attribute-Based Access Control** document the design and components prototyping of the attribute-based access-control framework. It assures implementation and integration of the cryptographic attribute-based authentication protocol to be used in conjunction with device-centric authentication.

The reference architecture is the basis for the design and the implementation of the technical solutions for ReCRED. The revised version of the reference architecture presents the holistic solution that is followed in ReCRED.

5 References

[1] “Why the Future of Identity is OpenID Connect and not SAML ”. http://apicrazy.com/2014/08/18/why-the-future-of-identity-is-openid-connect-and-not-saml/ as retrieved on 2015-11-28

[2] Brian McGillion and Tanel Dettenborn and Thomas Nyman and N. AsokanOpen-TEE- An Open Virtual Trusted Execution Environment, Proceedings of the 2015 IEEE Trustcom, Vol. 1, pp: 400-407, 2015

[3] Attribute-Based Access Control (ABAC) http://csrc.nist.gov/projects/abac/

[4] Camenisch, Jan, and Els Van Herreweghen. "Design and implementation of the idemix anonymous credential system", Proceedings of 9th ACM Conference on Computer and Communications Security. ACM Press, 2002.

[5] Camenisch, Jan. "Specification of the Identity Mixer Cryptographic Library, Version 2.3.1, December 7, 2010.

[6] Christian Paquin and Greg Zaverucha. “U-Prove Collaborative Issuance Extension”. Microsoft Research, June 2014.

[7] Christian Paquin and Greg Zaverucha. “U-Prove Cryptographic Specification V1.1”. Microsoft Corporation, April 2013.

[8] Christian Paquin. “On the Revocation of U-Prove Tokens”. Microsoft Research, September 2014.

[9] Christian Paquin. “U-Prove Technology Overview v1.1, Revision 2”. Microsoft Corporation, April 2013.

[10] Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In Advances in Cryptology – EUROCRYPT 2001, vol. 2045 of LNCS, pp. 93-118. Springer Verlag, 2001.

[11] FIDO alliance specifications overview. https://fidoalliance.org/specifications/overview/

[12] FIDO alliance UAF protocol specifications. https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-overview-v1.0-ps-20141208.html

[13] GlobalPlatform, “GlobalPlatform made simple guide: Secure Element ”, Media & resource Center, https://www.globalplatform.org/mediaguideSE.asp

[14] GlobalPlatform, “The Trusted Execution Environment - Delivering Enhanced Security at a Lower Cost to the Mobile Market”, June 2015 (revised from February 2011).

[15] Greg Zaverucha. “U-Prove ID Escrow Extension”. Microsoft Research, September 2013.

[16] Idemix back-end for the Credentials API. https://github.com/credentials/credentials_idemix

[17] IDEMIX Download. https://abc4trust.eu/index.php?option=com_content&view=article&id=187

[18] Introduction to Single Sign-On. http://www.opengroup.org/security/sso/sso_intro.htm

[19] JSON Web Token (JWT). https://tools.ietf.org/html/rfc7519

[20] LATCH. https://www.elevenpaths.com/technology/latch/index.html

[21] Mira Belenkiy. “U-Prove Equality Proof Extension”. Microsoft Research, June 2014.

[22] Mira Belenkiy. “U-Prove Inequality Proof Extension”. Microsoft Research, June 2014.
[23] Mira Belenkiy. “U-Prove Range Proof Extension”. Microsoft Research, June 2014.
[24] Mira Belenkiy. “U-Prove Set Membership Proof Extension”. Microsoft Research, June 2014.
[25] OpenID connect. http://openid.net/connect/
[26] Security Assertion Markup Language (SAML). http://saml.xml.org/
[27] The OAuth 1.0 Protocol. https://tools.ietf.org/html/rfc5849
[28] The OAuth 2.0 Authorization Framework. https://tools.ietf.org/html/rfc6749
[29] Trusted Computing Group, “TPM Main Part 1 Design Principles”, 2003 – 2011.
[30] U-Prove. http://research.microsoft.com/en-us/projects/u-prove/
[31] FiWARE. https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Privacy_Open_RESTful_API_Specification
[32] XACML Specification. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html