

IdeNtity verifiCatiOn with privacy-preservinG credeNtIals for
anonymous access To Online services

INCOGNITO

WP2 – Requirements, Architecture and Ethics
Deliverable D2.3 “Reference architecture”

Editor(s):	CUT
Author(s):	Kostantinos Papadamou (CUT– outside secondment), Markos Charalambous (CUT– outside secondment), Petros Papagiannis (CUT secondee), Michael Sirivianos (CUT– outside secondment), Christos Xenakis (UPRC– outside secondment), Vaios Bolgouras (UPRC secondee), Anastasia Tsiota (UPRC secondee), Dimitrianos Savva (LSTech secondee), Evangelos Kotsifakos (LSTech – outside secondment), George Gugulea (CSGN– outside secondment), Sorin Catalin Teican (CSGN– outside secondment), Petru Scurtu (CSGN– outside secondment), Ioana Stroinea (CSGN secondee), Katerina Samari (FOGUS secondee), Carlos Segura (TID– outside secondment)
Dissemination Level:	Public
Nature:	Report
Version:	2.0



INCOGNITO is funded by the European Commission’s Horizon 2020 Research and Innovation Framework program under the Marie Skłodowska-Curie Research and Innovation Staff Exchanges Action, Grant Agreement no 824015. The content of this deliverable reflects only the views of the project owner. The European Agency / Commission is not responsible for any use that may be made of the information it contains.







PROPRIETARY RIGHTS STATEMENT

This document contains information, which is proprietary to the INCOGNITO Consortium. Neither this document nor the information contained herein shall be used, duplicated or communicated by any means to any third party, in whole or in parts, except with prior written consent of the INCOGNITO consortium.

Project Profile

Contract Number	824015
Acronym	INCOGNITO
Title	IdeNtity verifiCatiOn with privacy-preservinG credeNtials for anonymous access To Online services
Start Date	Jan 1 st , 2019
Duration	48 Months

Partners

	Τεχνολογικό Πανεπιστήμιο Κύπρου	TECHNOLOGIKO PANEPISTIMIO KYPROU (BEN2)	CUT	Cyprus
	University of Piraeus	UNIVERSITY OF PIRAEUS RESEARCH CENTER (BEN1)	UPRC	Greece
	certSIGN®	CERTSIGN SA (BEN3)	CSGN	Romania
	Telefónica Investigación y Desarrollo	TELEFONICA INVESTIGACION Y DESARROLLO SA (BEN6)	TID	Spain
	LSTech	LSTECH ESPANA SL (BEN4)	LST	Spain
	FOGUS	FOGUS INNOVATIONS & SERVICES P.C. (BEN7)	FOG	Greece

Document History

VERSIONS

Version	Date	Author	Remarks
0.1	03/03/2020 (M15)	Kostantinos Papadamou (CUT), Markos Charalambous (CUT)	Executive Summary and Table of Contents
0.2	04/03/2020 (M15)	Kostantinos Papadamou (CUT), Markos Charalambous (CUT), Vaio Bolgouras (UPRC)	Introduction
0.3	09/03/2020-31/03/2020 (M15)	Markos Charalambous (CUT), Petros Papagiannis (CUT), Dimitrianos Savva (LST), Vaio Bolgouras (UPRC), Ioana Stroeina (CSGN), George Gugulea (CSGN), Vaio Bolgouras (UPRC)	Architecture Building Blocks
0.4	02/04/2020-30/04/2020 (M16)	Kostantinos Papadamou (CUT), Markos Charalambous (CUT), Petros Papagiannis (CUT), Ioana Stroeina (CSGN),	Protocols and Technologies

		Sorin Catalin Teican (CSGN), Petru Scurtu (CSGN), Vaios Bolgouras (UPRC)	
0.5	04/05/2020- 05/06/2020 (M17-M18)	Kostantinos Papadamou (CUT), Markos Charalambous (CUT), Petros Papagiannis (CUT), Katerina Samari (FOGUS), Carlos Segura (TID), Ioana Stroinea (CSGN), Petru Scurtu (CSGN), Vaios Bolgouras (UPRC)	Architectural Components
0.6	08/06/2020- 29/06/2020 (M18)	Michael Sirivianos (CUT), Christos Xenakis (UPRC), Evangelos Kotsifakos (LST)	Document Review
1.0	30/06/2020	Nikos Passas (UPRC)	Submission
2.0	31/12/2020	Nikos Passas (UPRC)	Submission of revised version

Deliverable dates

Delivery	Date
Contract delivery due date	30/06/2020 (M18)
Actual delivery date of version n.1	30/06/2020 (M18)
Actual delivery date of version n.2	31/12/2020 (M24)

Fellow ID	Name/Surname	Researcher category	Declaration No.	PM
1	Dimitrianos Savva	ESR	1	3
5	Petros Papagiannis	ESR	4	3
10	Katerina Samari	ER	8	3
11	Ioana Stroinea	ESR	9	3
16	Vaios Bolgouras	ESR	13	4
15	Anastasia Tsiota	ESR	12	0.25

Executive Summary

The current deliverable documents the reference architecture of the INCOGNITO project and is part of the Work Package 2. The development of the project is taking place under a partnership that involves both academic and industrial expertise. Considering the interdisciplinary knowledge exchange, the consortium involved in INCOGNITO aims to create a platform that leverages the state-of-the-art technologies so that users may securely, privately and effortlessly access online services, as well as to add features that guide users through the registration and authentication processes without disclosing their identity such as an AI-based assistant.

The purpose of this report is to define and describe the INCOGNITO framework architecture, its components, the interaction between these components and the technologies that will be used for the implementation of each component. The breakdown of the various components has also been based on the use cases (pilots) and the technical requirements (user stories) defined in Deliverable 2.1 of the project.

This document is organized as follows. The general INCOGNITO architecture together with a brief description of the project is provided in the introduction. In Section 2, an introductory description of each one of the five building blocks of the INCOGNITO platform (technologies used and architectural components involved) is presented: (a) User-to-device Authentication; (b) Qualified Anonymity; (c) Identity Acquisition and Management; (d) User-based Consent Management; and (e) Advanced UI/UX AI-based Assistant. In Section 3, we provide a detailed description of the various architectural components of the INCOGNITO platform: (a) User Device (UD); (b) Identity Consolidator (IDC); (c) Identity Provider (IdP); and (d) Service Provider (SP). We note that, the description of the INCOGNITO architecture in Section 3 follows a different structure than the one mentioned in the description of task T2.3 of the Grant Agreement. This is mainly because task T2.3 provides a more high-level description of the three main concepts/building blocks of INCOGNITO, while in this section it is important that we properly define and describe all the components that must be implemented in order to properly implement all the functionalities of INCOGNITO, while also following the specifications defined by the protocols that we utilize. Last, in Section 4 we provide a reference to all the technologies and protocols that will be used for the implementation of the INCOGNITO platform and we conclude in Section 5.

Table of Contents

Executive Summary	4
Table of Contents.....	5
List of Figures	6
Table of Abbreviations	7
1 Introduction	8
2 Authentication and Identity Management in INCOGNITO.....	8
2.1 User-to-Device Authentication	9
2.2 Qualified Anonymity	10
2.3 Identity Acquisition and Management.....	10
2.4 User-based Consent Management.....	11
3 Architectural Components (Architectural Building Blocks)	12
3.1 User Device (UD)	12
3.2 Identity Consolidator (IDC).....	14
3.2.1 Identity Acquisition Module	15
3.2.2 Account Management	20
3.2.3 Credential Management	23
3.2.4 Authentication Management Module	26
3.2.5 Identity Management Module.....	27
3.2.6 AI-based Assistant Backend	28
3.2.7 Tor Network.....	29
3.2.8 Blockchain	30
3.2.9 Identity Repository and Third-Party API.....	30
3.3 Identity Provider (IdP)	31
3.3.1 FIDO2 Server.....	32
3.3.2 QR Authentication Server.....	33
3.4 Service Provider (SP).....	35
4 Protocols and Technologies.....	35
4.1 FIDO 2	35
4.1.1 Overview	35
4.1.2 WebAuthn	37
4.1.3 CTAP2	40
4.1.4 FIDO2 integration with INCOGNITO Architecture.....	41
4.2 OAuth 2.0.....	42
4.2.1 OpenID Connect	42
4.2.2 User-Managed Access (UMA)	44
4.3 Privacy Attribute-based Access Control (PABAC).....	54
4.3.1 Idemix	55
4.3.2 Attribute-Based Encryption – OpenABE.....	60
4.4 Blockchain	60
4.4.1 Hyperledger Fabric	60
4.4.2 Ethereum	61
4.4.3 Quorum	61
4.4.4 R3 Corda	62
4.4.5 INCOGNITO blockchain Network.....	62

4.5	Near-Field Communication (NFC)	64
4.5.1	NFC Implementations	64
4.6	WebRTC	66
4.7	Tor Network.....	67
5	Conclusions.....	68
6	References.....	70

List of Figures

Figure 1. Overall INCOGNITO Reference Architecture.....	9
Figure 2. User Device Protocol Stack View	12
Figure 3. TEE and Open-TEE functionalities in INCOGNITO.....	13
Figure 4. Identity Consolidator Stack View	14
Figure 5. Identity Acquisition with Identity Consolidator and Verification Component View.....	16
Figure 6. User Interaction with Physical Identity Acquisition	17
Figure 7. Online Identity Acquisition Module	19
Figure 8. Credential Issuance using the Credential Management Module	24
Figure 9. Storage Model	31
Figure 10. User’s Device Protocol Stack View	31
Figure 11. Routing through gateSAFE	33
Figure 12. QR Authentication.....	34
Figure 13 Service Providers Protocol Stack View	35
Figure 14. FIDO Registration	39
Figure 15. FIDO Authentication.....	40
Figure 16. Authorization code flow	43
Figure 17. Implicit flow	44
Figure 18. UMA general flow.....	45
Figure 19. UMA Flow	47
Figure 20. UMA Architecture in WSO2 Server	50
Figure 21. Attributed-Based Access-Control component view.....	55
Figure 22. Mapping of INCOGNITO components to the roles of the Idemix Protocol ..	56
Figure 23. Idemix Credential Structure and Data	59
Figure 24. Attribute Structure inside a Credential	59
Figure 25. Blockchain network	63
Figure 26. Mobile ID solution and identity verification process	65

Table of Abbreviations

ABAC	Attribute-based Access Control
AI	Artificial Intelligence
CCS	Cryptographic Credentials Storage
CTAP2	Client to Authenticator Protocol
FIDO2	Fast IDentity Online version 2
IDC	Identity Consolidator
IdP	Identity Provider
LoA	Level of Assurance
NFC	Near-Field Communication
OIDC	OpenID Connect
SP	Service Provider
TEE	Trusted Execution Environment
UD	User Device
PABAC	Privacy-Preserving Attribute-based Access Control
PAT	Protection API Access Token
RPT	Requesting Party Token
RFID	Radio Frequency Identification
UMA	User-Managed Access
UX	User Experience
WebRTC	Web Real-Time Communication

1 Introduction

The INCOGNITO project is built upon the results produced by the ReCRED project, and the added value of this project comes from enhancing existing functionalities and utilizing upgraded technologies.

In addition to the password-less authentication experience and the preservation of users' privacy, in INCOGNITO platform the User Experience (UX) is considered as a high priority. To this end, an AI-based assistant will be available to the user in order to assist with actions that need to be taken regarding identity management and interaction with the service providers. Another feature brought by the INCOGNITO project is the last FIDO specification¹, namely FIDO2, which offers to the user the ability to use external authenticator factors (such as secure USB tokens), not just biometric information that can be collected using the user's device.

Furthermore, INCOGNITO offers an enhanced Identity Acquisition and Management framework. First, the collected and verified identity attributes are stored on the blockchain, rendering them available for verification by authorized parties, as well as secure and tamper proof². The blockchain technology will also be utilized to track changes regarding granting and gaining access to user identity attributes, in order to avoid disputes and log information that may be needed when conducting an audit or investigation. User-Managed Access (UMA) protocol will also assist in this, giving the user complete control over the policies he sets and over who may have access to the corresponding identity attributes[1].

2 Authentication and Identity Management in INCOGNITO

In this section, we introduce the INCOGNITO reference architecture, and we provide an overview of how authentication and identity is realized in INCOGNITO. We also describe how the different building blocks of the INCOGNITO framework communicate with each other in order to achieve user authentication and guidance through the client application, while granting access to services by sharing only the required identity attributes with the Service Providers.

Figure 1. **Overall INCOGNITO Reference Architecture** presents the reference architecture of the INCOGNITO platform, which includes all the main components and the interconnections between them. The protocols and technologies that will be used to implement this architecture are described in Section 4. The detailed specifications of the protocols and the components of the architecture will be described in detail in the corresponding work package deliverables.

¹ <https://fidoalliance.org/specifications/overview/>

² <https://lup.lub.lu.se/student-papers/search/publication/8919957>

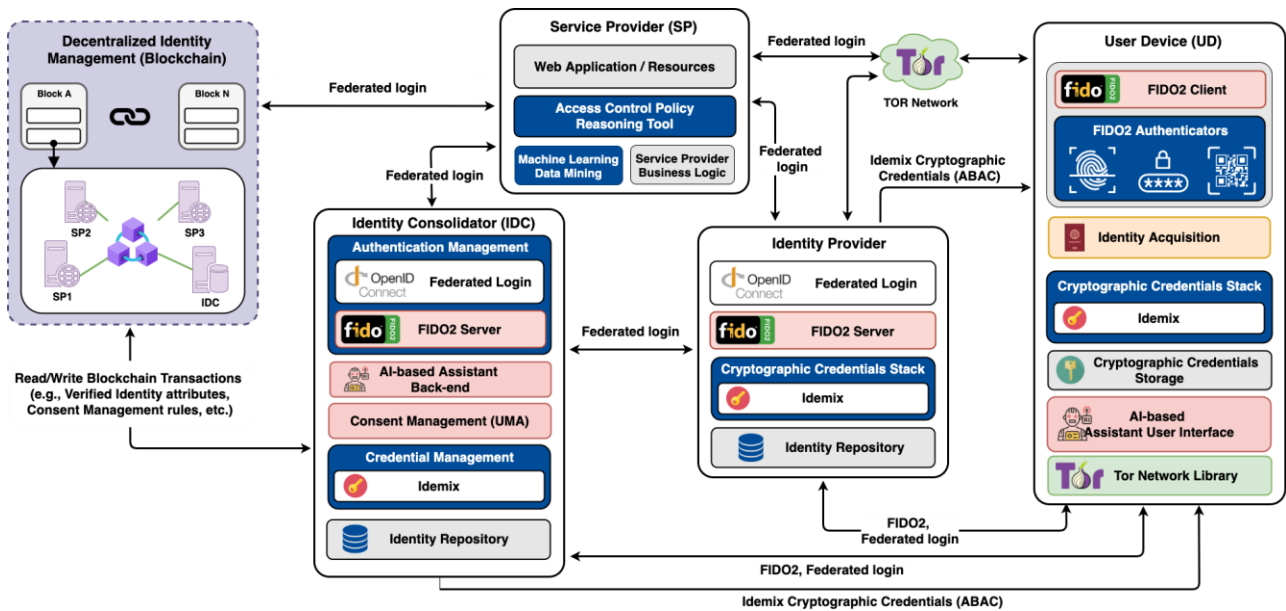


Figure 1. Overall INCOGNITO Reference Architecture

2.1 User-to-Device Authentication

The User-to-Device Authentication is the block which triggers the whole flow of the INCOGNITO platform. Each authentication at the Identity or Service Providers, each update a user makes regarding his identity at an online service, each transfer of identity attributes between entities are related to the user authentication to his/her own device. This is how all the interactions between parts of the system are securely taking place.

INCOGNITO takes advantage of the technologies available on the modern mobile phones, that are described in this paragraph. The user interacts with the user device through the device specific secure authentication mechanism, such as biometrics (face recognition, fingerprints) or a pin. These pieces of the user’s characteristics are converted in cryptographic keys that are securely stored on the user’s device. For this purpose, the device comes with a module called Trusted Execution Environment (TEE) which is an integration between a Trusted OS and hardware features that work together to accomplish the necessary security requirements. The main focus is set on ensuring that the user’s characteristics stored on the user’s device will not leave the device even in the case of a failure of the OS. On top of the TEE, a Cryptographic Credential Storage (CCS) runs, and it is there that the generated keys are stored on the user’s device.

INCOGNITO allows the user to authenticate to their device through roaming authenticators whose usability is enabled by the use of the FIDO2 protocol. These authenticators are connected to the user device when necessary and they are being interacted with by the user in order to prove that they own the connected authenticator. This authentication method is described in more detail in the Protocol and Technologies chapter - FIDO2 subsection.

There is one main requirement of the user-to-device authentication regarding security: INCOGNITO has to make sure that the user’s characteristics never leave the user’s device and that any further transfer of information is authorized through cryptographic protocols that use cryptographic keys generated from the user’s characteristics. In this way, the user’s physical attributes are kept secure and the user is not in danger of being subject to malicious use of physical personal data.

2.2 Qualified Anonymity

We ensure “qualified anonymity”, which means, that the online service does not have any information about the user but a pseudonym. In order to achieve that we incorporate federated login solutions such as OpenID Connect³ (OIDC) with anonymous credential systems such as Idemix[2] or OpenABE⁴. This allows the Online Services to receive a limited subset of the user’s identity attributes that will guarantee that the user is qualified to access a resource. Cryptographic credentials can be used by the user in order to prove ownership of an identity attribute; the cryptographic credentials are submitted to the Identity Provider which utilizes the corresponding cryptographic credentials stack (Idemix or OpenABE), and ultimately the identity attributes are communicated to the Service Provider using OIDC. At the same time the user will remain anonymous, hence achieving the desired concept of Qualified Anonymity. Without that solution, the users would have to share their identity in its entirety with the service providers, in order to utilize the services, they want. TOR⁵ network is also utilized in order to boost anonymity when identity attributes are communicated through utilization of cryptographic protocol stacks like Idemix between the users and the Identity consolidator. This building block is further detailed in Section 4.3.

2.3 Identity Acquisition and Management

The Identity Acquisition can take place either physically or virtually (online). In order to cover both of these cases, two corresponding modules will be created, the Physical Identity Acquisition module and the Online Identity Acquisition module. The utilization of TEE[3] on the devices used to acquire the identity attributes contributes to our goal of meeting the requirements set by the GDPR⁶, ensuring the security of the identity attributes from the design phase of the platform. The integration of blockchain technology adds to that, by ensuring that the users’ identity attributes cannot be altered by unauthorized entities.

The Physical Identity Acquisition module is responsible to acquire and verify the identity attributes included in a user’s real-world identity (such as passport, national ID card, driving license, etc.). A Near-Field Communication[4] (NFC) protocol will be integrated to easily, quickly and securely acquire the identity attributes from the end-user RFID-enabled physical ID documents (e.g., eID, e-Passport, etc.). In addition, the WebRTC⁷ protocol will be integrated in INCOGNITO, and will

³ <https://openid.net/connect/>

⁴ <https://github.com/zeutro/openabe>

⁵ <https://www.torproject.org>

⁶ <https://gdpr-info.eu/>

⁷ <https://webrtc.org/>

be enhanced with the required features so that we are able to offer remote identity verification. The process that an end-user has to follow is separated into two parts the acquisition and the verification of the identity attributes. On top of that, user identities that reside in the cyber space belong mainly to IdPs related to social networks (e.g., Facebook, LinkedIn). The goal of this module is to obtain the identity attributes of the user from such IdPs and consolidate that information to the IDC. The online identity acquisition module has two main processes, the acquisition of the identity attributes, and their integration to the user’s profile.

2.4 User-based Consent Management

User-based consent management is one of the basic pillars of the INCOGNITO platform since its main goal is to offer security and privacy to its users. INCOGNITO not only secures identity transfers or protects users’ privacy, but it also enables the users to gain more control and flexibility upon their shared identity attributes.

The protocol that enables user-based consent management in INCOGNITO is User Managed Access (UMA). The protocol relies on the use of an authorization server which is in charge of all the preferences set by the user regarding his identity. UMA enables users to define policies at the authorization server which will then be used asynchronous to the users’ online presence in order to allow access to the identity attributes. These policies offer a great spectrum of flexibility since the user can choose to share a certain attribute only with a certain Service Provider or can create different policies regarding different identity attributes with different degrees of access. Using UMA, the user takes control over their identity and can restrict the control of the Identity Consolidator over their own identity attributes. For example, if the Identity Consolidator has the ability to share name, age and address with a certain Identity Provider according to a degree of trust, the user can restrict the Identity Consolidator to only share their age with the respective Identity Provider. As an additional feature coming to guide the user in managing his identity, the INCOGNITO project also bounds the User-based Consent Management to an AI-based assistant which will have the necessary knowledge so that it can inform the user about the information needed to be disclosed to a certain entity and the risks the user is exposed to. As a result, the user can exercise their authority upon their identity management by means of giving their consent regarding the disclosure of their requested identity attributes. The communication between the user and the AI-based assistant is mediated by a highly intuitive User Interface. The user interface manifest either as a web interface or a mobile application. Furthermore, UMA will be enhanced by being integrated with ABAC. Together they will provide fine-grained authorization capable of dealing with dynamic and complex access relationships scenarios.

The technical way according to which user-based consent management manifests in INCOGNITO is described in more detail in the Protocol and Technologies chapter, User Managed Access section. There are described all the components necessary to implement the user-based consent management and explained by means of figures. The section goes even more into detail describing the actual implementation of UMA that INCOGNITO is going to use and the whole process of selection for the most suitable UMA server for the goals and needs of the project.

3 Architectural Components (Architectural Building Blocks)

In this section, we describe the reference architecture of the INCOGNITO platform focusing on the main architectural components.

The main components of the INCOGNITO architecture are the following:

1. User Device (UD)
2. Identity Consolidator (IDC)
3. Service Providers (SP)
4. Identity Providers (IdP)
5. Decentralized Identity Management

3.1 User Device (UD)

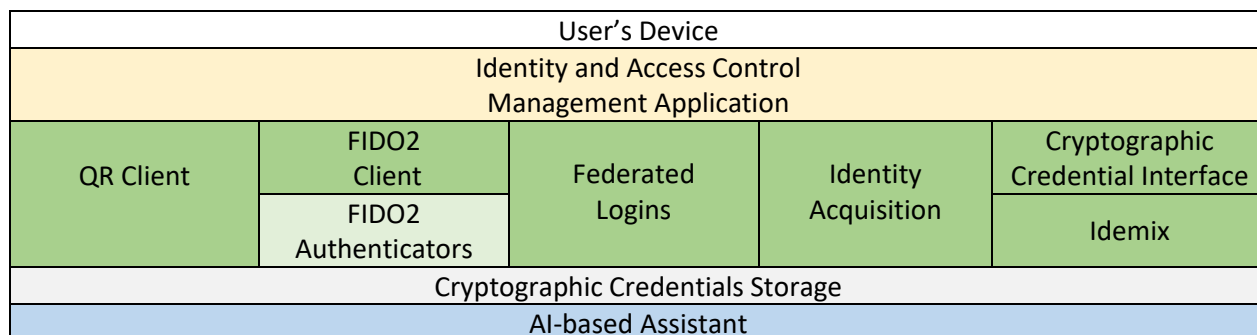


Figure 2. User Device Protocol Stack View

In INCOGNITO, the user mobile device is one of the main components as we intend to achieve a device-centric authentication. In order to achieve the authentication phase between the user and a remote service, the FIDO2 protocol is used. This module consists of the FIDO2 Client and the FIDO2 Protocol stack which both run on the user device and communicate with the Relying Party. Furthermore, INCOGNITO supports Attribute-Based Access Control (ABAC)[5] cryptographic credentials to Identity Providers.

The user's cryptographic credentials, which are received from the Identity Consolidator and the multiple IdPs, are stored in the cryptographic credentials storage that is allocated in the user's device. Also, to enhance the security of the credentials of the user, the cryptographic credentials storage takes advantage of the user device's Trusted Execution Environment (TEE) capabilities. A cryptographic interface is also included in the user device which communicate with the Idemix protocol stack that are installed on the user's device. The Idemix credentials that are saved in the Cryptographic Credential Storage can be released from these stacks.

To enhance the TEE capabilities of the user device we use the Open-TEE⁹ which is an open-source project where we can develop critical functionalities for anonymous credential protocols (Idemix for INCOGNITO).

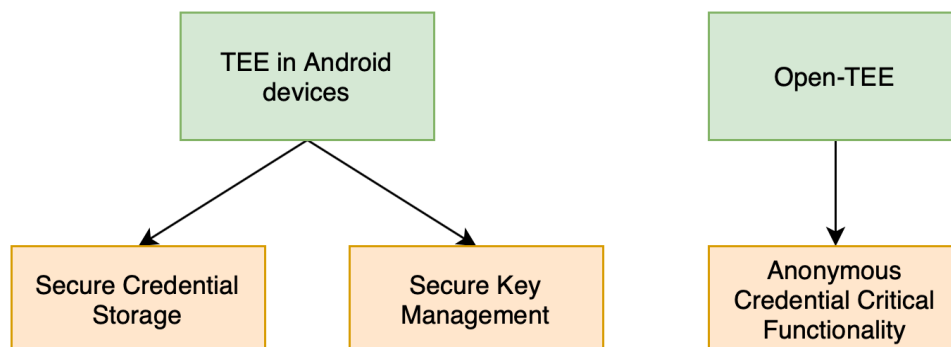


Figure 3. TEE and Open-TEE functionalities in INCOGNITO

For authentication and authorization purposes between the IdP and SPs, the user device runs federated login protocols (OpenID Connect/OAuth 2.0). In addition, we integrate FIDO2 for authentication between the user device and the Identity Provider, which supersedes the standard password paradigm.

The user has the option to control his identity information with the identity and access control management application, which is also included on the user device and communicates with the IDC. The application enables the user to manage his identity attributes exposed to each Service Provider. The user can also issue cryptographic credentials from his identity attributes directly to his device and use them for ABAC. The consent management is also included in the application and ID privacy functionality that enables the users to have knowledge and control over which Service Provider knows which aspects of his identity.

On the user device there is also an AI-based Assistant, which communicates with the Identity Consolidator to inform the user, as well as to guide him into properly managing his/her identity. The AI-based Assistant notifies the user, for example, about the minimum identity attributes required to be revealed to the Service Providers in order to get access to its resources. Also, the AI-based Assistant has the ability to inform the user about the risk of revealing specific identity attributes to Service Providers which in turn may enable the SPs to infer complete identity of the user.

The device has an identity acquisition module that will allow the user to quickly and securely acquire identity attributes from online identities (Facebook account, etc.), as well as his/her physical ID Documents (e-Passports, eID, etc.) with the Near-Field Communication (NFC) protocol. The user will have the option to store the acquired and verified identity attributes to the IDC.

The QR client is a module which is also included on the user device for transferring the credentials. The user can scan a QR code to access the Service from his browser while he already authenticated using his mobile device. The Identity Provider uses the QR code to verify that the user is authenticated and permits the user to create a new session without the need for re-authentication.

3.2 Identity Consolidator (IDC)

Identity Consolidator						
Identity and Access Control Management Application						
Identity Acquisition and Verification		Consent Management	Authentication Management		Account Management	Credentials Management
Verification Algorithms	Federated Logins	ID attributes interference algorithms	QR Auth. Server	FIDO2–enhanced Keycloak	LATCH	Idemix
				FIDO2 Server		
Identity Attribute Store		AI-based Assistant Back-end	Identity Consolidator Data			

Figure 4. Identity Consolidator Stack View

The Identity Consolidator is the central node of INCOGNITO, integrating different modules described in the following sections. Firstly, the Identity Consolidator is the place where the online identity of the user lies. It has modules that gather real-world and online information about a user and makes one central trusted identity. After transposing this user information into identity attributes at the Identity Consolidator, the latter acts as a connection point between the user and Identity or Service Providers. It mediates and makes trustworthy the communication needed for a user to get access to a Service Provider’s resources by generating cryptographic credentials for the users in order to authenticate at different Service Providers or by hosting different servers such as FIDO2 for secure authentication or UMA for user-based content management. It also enables the user to manage their own identity through modules such as Account Management Module or Identity Management Module. Therefore, the user can enhance the security of their account and can also restrict the control of the Identity Consolidator over their identity by making use of the UMA server. The user can set policies that have a higher priority over the identity attributes than the level of assurance calculated by the Identity Consolidator at the moment of the integration of the information gathered from different Identity Providers about the user’s identity.

The Identity Consolidator also enables the user to make use of the services of an AI-based assistant that will assist the user in the management and disclosure of their identity. The AI-based assistant will have the expertise to prevent the user about potential risks and allow them to take actions regarding this matter, as well as being also in charge of taking actions itself in order to protect user’s privacy and security.

Two different layers of security are added by using a Tor Network to mediate the communication between the user and the Service Providers and blockchain to log user’s interactions with the online services such as Identity Consolidator or Service Providers.

Since the Identity Consolidator was a component in the ReCRED project too, many of the basic components such as Account or Credential Management are also detailed in the respective project.

Among these modules, other new features and technologies were also added in this deliverable such as a FIDO2 server, a UMA server, the back-end software stack of the AI-based assistant and blockchain technology.

3.2.1 Identity Acquisition Module

The Identity Acquisition Module performs vertical and horizontal binding of the user’s identity. It contains two modules which bind the real-world identity and the various online identities into one trusted online identity stored at the Identity Consolidator. The first module – Physical Identity Acquisition Module performs a vertical identity binding of the real-world characteristics of the user to the online identity attributes. The second module – Online Identity Acquisition Module performs horizontal binding between different digital identities of a user stored across multiple online sources such as social networks (Facebook Identity, Google Identity etc.). Together, the two modules create a trusted online identity which abstracts the real-world identity of the user through processes that perform reliable and secure validation of the match between the user’s real-world identity and their online identity attributes.

3.2.1.1 *Physical Identity Acquisition Module*

The Physical Identity Acquisition Module binds a user’s real-world identity with the online identity through information gathering from physical proofs such as passport or identity card. The real-world identity is transposed in the online world as verifiable identity attributes saved at a trusted provider. The means through which the attributes are collected are an application implemented on smart trusted-computing-enabled devices and a web application.

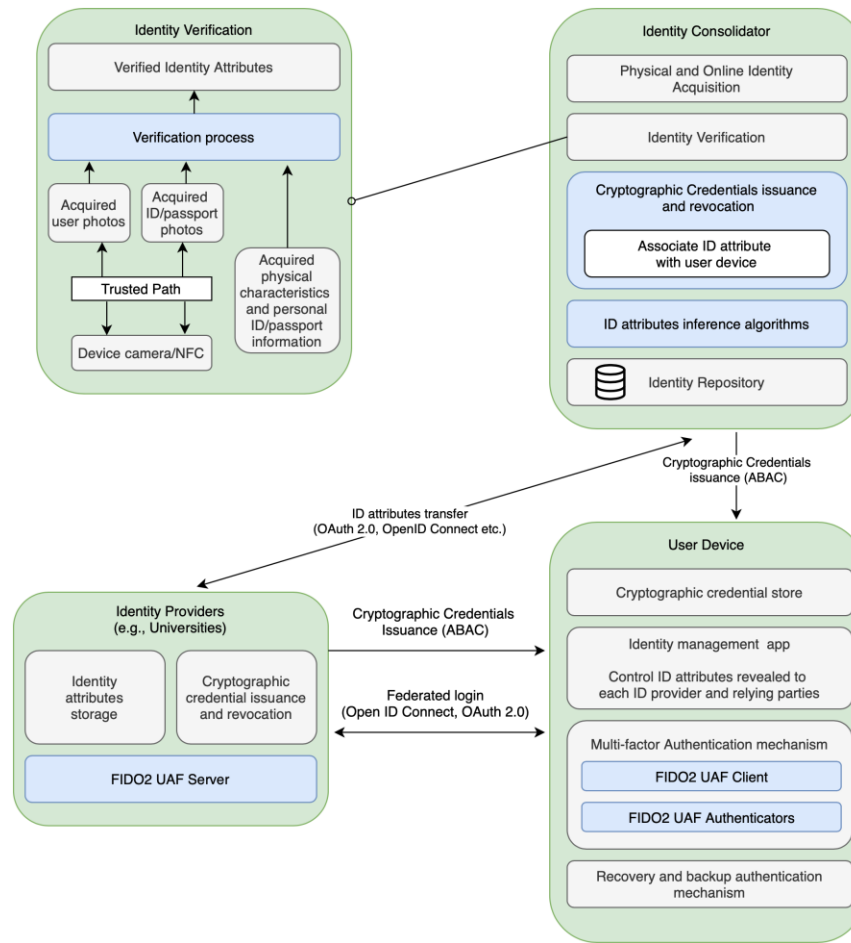


Figure 5. Identity Acquisition with Identity Consolidator and Verification Component View

The Physical Identity Acquisition consists of two steps: The Identity Acquisition Process and the Identity Verification Process which will be detailed in the sections below.

The Identity Acquisition Process is in charge of gathering physical characteristics of the user or physical identity documentation using the user’s device. The acquisition is mediated by sensors located in commodity mobile devices (fingerprint scanner, camera for face recognition, GPS for location) or communication protocols (i.e., NFC to scan ePassports, WebRTC for remote identity verification) and the transmission of the received information takes place through trusted software paths. The following step is to transfer all this data to the Identity Verification Process which will verify the acquired characteristics through crowdsourcing and automated techniques acting along with user security and privacy preserving. The following figure explains how the user interacts with the Physical Identity Acquisition Module.

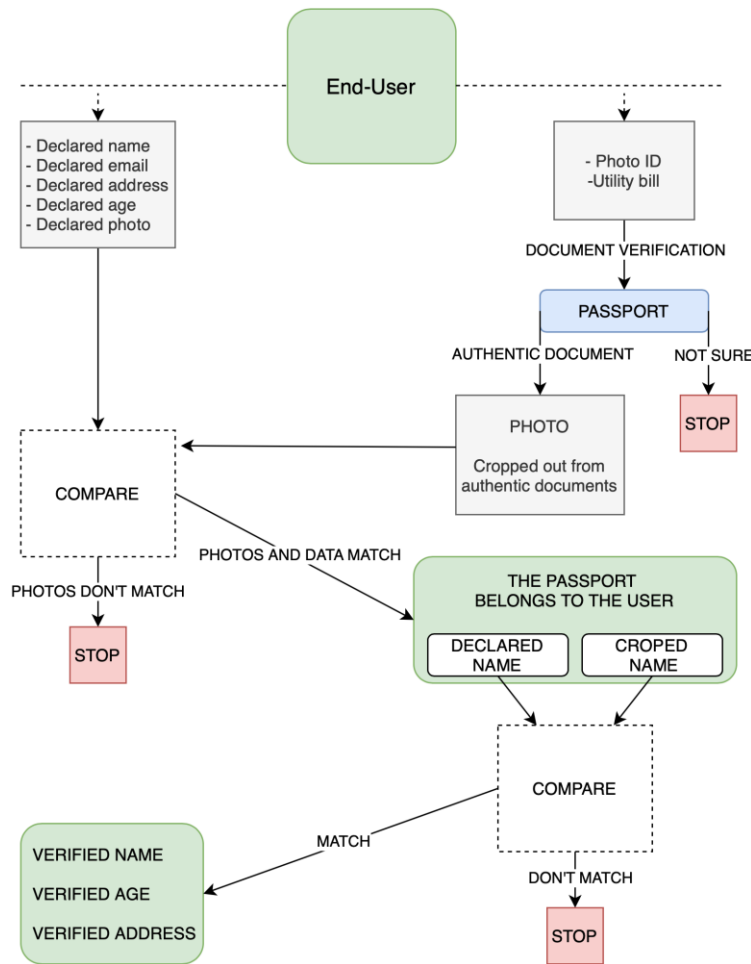


Figure 6. User Interaction with Physical Identity Acquisition

A. Identity Acquisition Process

For the Identity Acquisition Process, the INCOGNITO project provides the user with a web or mobile interface which will enable the user to declare their identity information such as name, address, identity card number etc. The information gathering also involves storing data from the physical documents which are scanned through NFC or captured by the user with the camera. Further, the extraction and verification of the desired identity attributes (i.e., date of birth, age, name) takes place and it usually involves the user too. The user has the role to crop the required information from the images taken with the camera.

B. Identity Verification Process

The Identity Verification Process receives all the information gathered through the Identity Acquisition Process and it uses peer-to-peer verification and automated means in order to securely verify identity information. The techniques used are described below:

- A. Peer-to-peer verification** is a crowdsourcing technique used in INCOGNITO with the purpose of verifying the validity of the acquired user personal data and physical characteristics. A user first declares personal information through the Identity Acquisition Process and then he provides real-world proof of the information declared such as identity card photos. Through the peer-to-peer technique, the platform crowdsources the acquired information to other users of the platform who will verify the compatibility between the provided proofs and the declared information. So that during this process no one has access to reusable or forgeable photos of the user’s documents, watermarking and cropping techniques are involved.
- B. Face Detection and Recognition** is used to check that a user’s photo matches their identity card photo. The technique first identifies the presence of a face in an image and in case there is one, the recognition of the user’s face follows.
- C. Optical Character Recognition (OCR)** is used to identify and check user data. When a user crops parts of an image corresponding to certain pieces of information, OCR technique verifies if the cropped parts contain any characters and if so, they will be checked for compatibility with the information introduced by the user.

As soon as the information has been checked, it will be stored as independent identity attributes in the Identity Repository of the Identity Consolidator component.

3.2.1.2 *Online Identity Acquisition Module*

The Online Identity Acquisition Module is in charge of integrating multiple online identities of the user into one. The user has to accept to integrate other online identities at the Identity Consolidator. All these online identities lie at different Identity Providers such as Google or Facebook and will be gathered through OpenID Connect / OAuth or Facebook Connect at the Identity Consolidator which will further verify the validity of the acquired information by means of the Identity Integration Module and then store it.

A. *Online Acquisition Process*

The Online Acquisition Process binds a user’s online identities into one that is stored at the Identity Consolidator. So that the Identity Consolidator could be able to gather information from a certain Identity Provider, the user first has to authenticate at that Identity Provider and then attributes such as name, date of birth, address will be extracted, verified and stored in the Identity Repository from the Identity Consolidator. The figure below describes how the Online Identity Acquisition Process works:

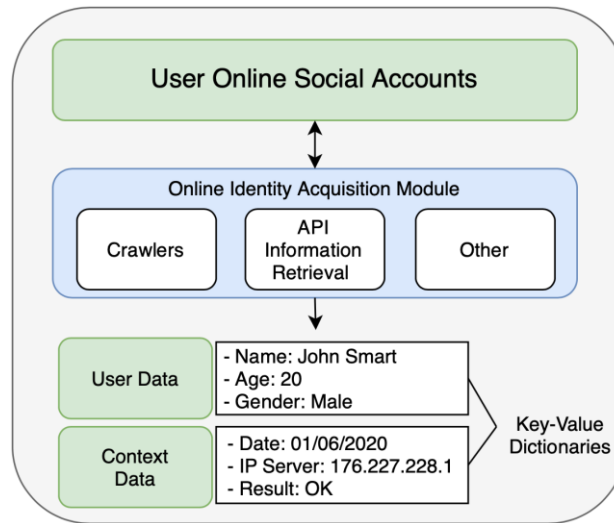


Figure 7. Online Identity Acquisition Module

B. Identity Integration Module

Since a user has digital identities spread across multiple sources such as social networks, blobs, portals or systems, they are managed and stored separately. Therefore, it is necessary to standardize and normalize the information in order to unify it into one identity. The Identity Integration Module provides exactly the integration between these different pieces of information in three main steps: data extraction, data transformation and data storage. The data extraction will happen through the Identity Provider API or by the non-user assisted module using web crawlers. The following steps are to transform the information and store it in the database by calling the Identity Attributes Storage API.

The Identity Integration module gathers together and connects all the acquired characteristics of the user and verifies their validity through means of statistical data analysis techniques. This module can also deduce missing identity attributes and is responsible for assigning confidence scores for the accuracy 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%. The considerations preceding the results are:

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 Identity Consolidator 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 penalization 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 three 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 two 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 penalization factor similarly as explained in the previous case.

3.2.2 Account Management

The Account Management Module is in charge of the online accounts of the user and the way they are exposed to authorized parties. It enables the users to register an Identity Provider with the Identity Consolidator in order to allow the IDC to extract data about its online identity from the Identity Provider. The Account Management Module is in charge of the user's account recovery, storing an IdP's URL and the corresponding username.

The process for account recovery contains the following steps:

1. If the Credentials Management Module does not have the secret key for the Identity Provider for which the account recovery takes place, then it initiates the Identity Provider's specific account recovery procedure.

2. If the Credentials Management Module has the secret key for the Identity Provider for which the account recovery takes place, then the Account Management Module retrieves the secret key and stores it on the new device. In this case the Identity Provider is left out of the recovery procedure.

The functionalities provided for the user by Account Management Module are:

1. **Identity Provider Registration**: The user can register and Identity Provider with the Identity Consolidator.
2. **User account deletion**: The user can delete their account from the Identity Consolidator.
3. **Account Recovery**: For the account recovery process, the user is presented with a list of the accounts he previously registered at the Identity Consolidator from which he can choose for which one to recover the secret key in order to save it on another device.
4. **Latch / Unlatch online accounts**: The user can access his list of online accounts and latch (lock) or unlatch (unlock) them. He can also set policies to automatically latch or unlatch accounts.
5. **View history of latch changes**: The user can see the history of the accounts and if they were locked / unlocked by himself or the Identity Consolidator through the use of policies.
6. **Define account locking policies**: A user can define policies in order to automatically lock or unlock accounts. For example, if a user decides to lock his email account during weekends. The Identity Consolidator has the ability to lock / unlock a user's account if it decides that there is a high risk of account compromise.
7. **Configure a Degree of Privacy**: When a user wants to increase or decrease the level of privacy to which his account is exposed to the Identity Consolidator, he can choose between a highest or lowest degree of privacy.

There are two types of degree privacy in INCOGNITO's Identity Consolidator:

1. **Highest degree of privacy**: In this case, the Identity consolidator acts as a discovery service under a Federated Identity model. It will only save the name, surname, the identity document number of the user and the Identity Provider where a certain attribute of the user is stored. Whenever a Service Provider needs that specific attribute, the Identity Consolidator reveals to the Service Provider the Identity Provider where it will be able to find the identity attribute it needs to allow access to its services.
2. **Lowest degree of privacy**: In this case, the Identity Consolidator will store all of the user's identity attributes and the credentials used to access external Service Providers. For this feature, the Account Management Module interfaces with the Identity Profile Management and the Credential Management module.

3.2.2.1 Identity Federation

The Account Management Module along with the Identity Profile Management provide Federated Identity services in INCOGNITO. When a Service Provider needs some attributes, but they do not exist at the respective Identity Provider, the latter queries the Identity Consolidator for the missing attributes. If the Identity Consolidator has the attributes stored on the Identity Repository, it will answer with the identity attributes themselves, otherwise it will redirect the Service Provider to the Identity Provider who owns the requested attributes. Regarding the Federated Identity services, the Identity Consolidator acts as a discovery service for the source of the requested attributes.

The Identity Consolidator has a complete list of the identity attributes of a user and the Identity Providers where they are found. If the Identity Consolidator itself does not know the needed attributes, then it will redirect the Service Provider to the Identity Provider owner of the attributes. If the user has set his account’s privacy degree as the highest, then the Identity Federation services will be the only way the Identity Consolidator interacts with the user and the Identity Providers. In case of a lowest degree of privacy, the Identity Consolidator either knows all of the user’s identity attributes or it can also act just as a discovery service since the user may opt not to allow the Identity Consolidator to transfer all his attributes from the Identity Provider.

3.2.2.2 Latch

In INCOGNITO, the functionality of a Latch is to temporarily enable or disable user accounts at third-party services. Despite having this functionality, it does not replace the identity management service at Service or Identity Providers, but merely adds an extra layer of security on top of the authentication layer. A third-party server must still implement and manage its own mechanisms to authenticate users.

So that a user could use the latch mechanism on his account, he must first create an account at the Latch Server and then pair his third-party accounts to the Latch account. In order to successfully complete this step, the third-party server must support latch. After the pairing process, all the accounts of the user at the Latch Server can be latched or unlatched at will. While an account is latched, it cannot accept authentication attempts from the respective user until it is unlatched again. The process takes place through the HMAC-SHA1 algorithm. A third-party server must register its application with the Latch Server and then receive its *applicationId* that identifies it at the Latch Server and the *applicationSecret* that is used to authenticate requests made to the Latch Server. On the other hand, the users must also create a Latch account and receive a *userId* and a *userSecret* that is used to authenticate requests made to the Latch Server.

The most common operations supported by the Latch functionality are described in the following list:

1. Account Pairing: Account pairing takes place after a user has received a 6-char random string token from the Latch Server. The token will be sent to the third-party server and will be exchanged with the Latch Server for a 64-char *accountId* that represents the third-party account at the Latch Server.
2. Check Status: Using the *accountId*, a third-party server has the ability to query a latch server about the status of its account (check if the account is latched or unlatched).

3. Change Status: A user can change the status of their third-party accounts at the latch server.
4. Latch Support Tool: Considering the goals of the INCOGNITO project, the latch functionality will be adapted to secure users’ data. An Account Locking Mechanism will be implemented in order to allow the Identity Consolidator to latch all of the accounts of a user in case of theft or other identity disclosure risks.

3.2.3 Credential Management

The Credential Management Module in INCOGNITO relies on the integration of the Idemix cryptographic engine to issue cryptographic credentials on the user devices. This module will be an extension of the cryptographic functionalities provided by the INCOGNITO credential issuance module running at the Identity Providers. The issued credentials include a set of cryptographic attributes which are either stored at the Identity Consolidator or managed by a third-party Identity Provider. If the third-party Identity Provider supports the issuance of cryptographic credentials, then the user device will be redirected to the respective Identity Provider when requesting credential issuance, otherwise the Identity Provider will rely on the Identity Consolidator’s cryptographic credential issuance.

There are two main challenges regarding the development of the Credential Management Module:

1. To securely map the identity attributes acquired from the Identity Consolidator to cryptographic credentials which will be issued on the user’s device.
2. To provide interfaces and functionalities to enable the credential issuance from different sources.

In order to start the credential issuance process, the user has to be authenticated with his personal device which is further authenticated with the Identity Consolidator. Another alternative is to authenticate with the web interface provided by the Credential Management Module in order to start the credential generation process. The user should trust the Identity Consolidator since it is in charge of the matching checks between the identity attributes and the requested credentials before the actual issuance starts. In the end, the generated credentials are stored on the user’s device and could also be backed up on the Identity Consolidator if the user chooses so. The Identity Consolidator must offer strong encryption in order to secure the backup process of the user’s issued credentials. The following figure provides the high-level architecture of the Credential Management Module.

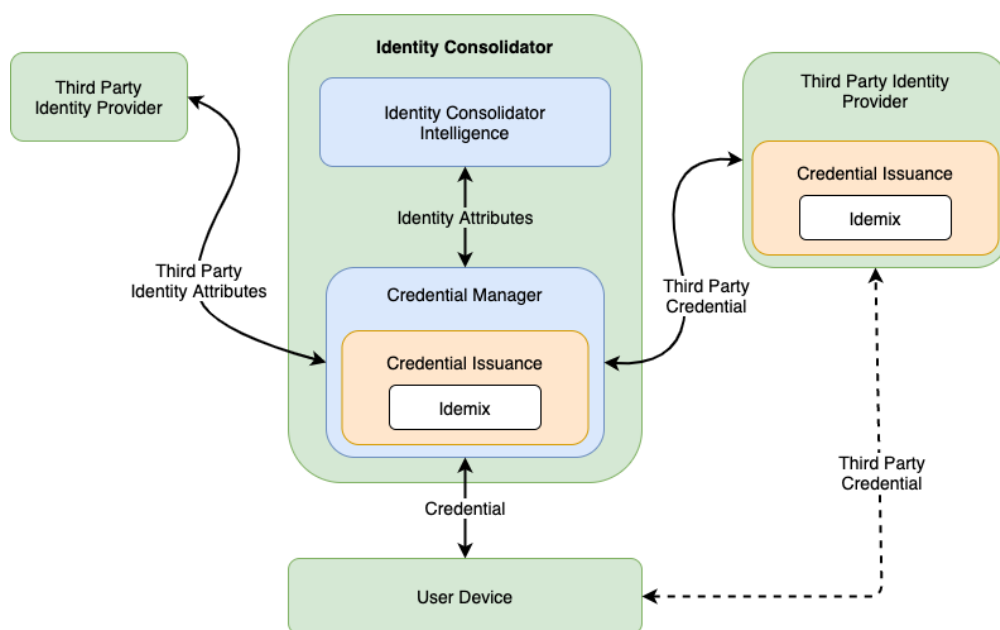


Figure 8. Credential Issuance using the Credential Management Module

The ways in which the Credential Management Module enhances the Identity Consolidator so that it could support cryptographic credential issuance are:

- The Credential Management module may serve as a trusted Identity Portal for External Identity Providers supporting INCOGNITO 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 INCOGNITO credentials issuance. The user contacts the Identity Consolidator in order to request a credential from an Identity Provider, as above. In this case the IDC 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 INCOGNITO issuance module and the user.
- Identity Providers that do not support the INCOGNITO 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 INCOGNITO 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 Credential Management Module operations exposed to the users are:

1. Issue cryptographic credentials: INCOGNITO supports different credential formats which can be generated either by the Identity Consolidator or by the Identity Provider. There are times when the credentials can be automatically issued in order to grant access to an online service, according to specific rules defined by the provider.
2. List supported attributes per IdP: The user has access to a list of all the Identity Providers and the attributes they support.
3. List issued cryptographic credentials: The user can access a list of all the credentials that have been issued to their device.
4. View issued cryptographic credentials’ details: The user can access more details upon selection of an issued credential, such as issued date, expiration date etc.
5. Reissue expired cryptographic credentials: The user can select an expired credential in order to renew it. The user can also set the Identity Consolidator to automatically renew an expired credential.
6. Encrypt / decrypt cryptographic credentials: The user can select one or more credentials in order to encrypt or decrypt them.
7. Backup cryptographic credentials: The user can opt to save the credentials at the Identity Consolidator. The transfer of the credentials from the user device to the IDC takes place through a secure channel. The user can enable automatic backup of the credentials at the Identity Consolidator.
8. Restore cryptographic credentials: A user can restore on his device the credentials backed up at the Identity Consolidator.
9. Erase cryptographic credentials: The user can select which credentials to erase and he can choose to delete them from the user device, Identity Consolidator or both.

The supported Credential Management Module operations for the Identity Providers are:

10. Manage supported identity attributes: The Identity Provider administrator is presented with all the identity attributes needed by various credential templates and choose which ones are supported by the Identity Provider. In order to generate a credential, it is necessary that the Identity Provider support all the identity attributes used by the credential template.
11. Issue cryptographic credentials: An Identity Provider administrator can issue credentials manually which may or may not have an expiration date. The credentials will be sent to the user’s device.
12. List issued cryptographic credentials: The Identity Provider administrator can access a list of all the credentials that have been issued.
13. Revoke cryptographic credentials: The Identity Provider administrator can revoke a specific credentials. As a consequence, the user will not be able to use it anymore.

14. View statistics: The Identity Provider administrator can see statistics regarding cryptographic credentials issued by the IdP.

3.2.4 Authentication Management Module

The Authentication Management Module offers a variety of authentication methods which are mainly designed to work according to a certain Level of Assurance (LoA). Each level of assurance is bound to a certain authentication error. As the consequences of an error increase, the level of assurance increases too. The supported authentication methods comply with the Levels of Assurance defined by the National Institute of Standards and Technology (NIST) and with the errors specified for each LoA. Each LoA allows token methods for all of its lower levels.

INCOGNITO offers the user a mobile application for the Authentication Management Module. The user registers with the Identity Consolidator up to a Level of Assurance which depends upon the proof of his identity that the user has provided and whether his device or the Identity Providers support the appropriate soft or hardware cryptographic tokens. As part of the registration, the Identity Consolidator issues FIDO credentials on the user’s device. The registration can also happen through federated means. A user by means of an Identity Provider will provide through OpenID Connect proof to the Identity Consolidator that he has an account with the respective LoA at the Identity Provider. It is mandatory for the IDC to trust the Identity Provider for the given LoA in order to accept a registration through federated identities.

The user registers and authenticates at the Identity Consolidator using FIDO credentials issued by the Identity Consolidator. The LoA to which a user authenticates at the IDC depends on the authentication method which is decided by the Identity Provider according to the NIST guidelines. For example, if the user authenticates to the IDC solely by using their 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. As soon as the user is authenticated, he can alter attributes (view, transfer, delete) according to the LoA level. The Identity Provider can set policies to restrict user’s access to attributes depending on his session of LoA. Once the user is authenticated to both the Identity Consolidator and Identity Provider, he can transfer attributes from the Idp to the IDC having the minimum LoA of the two different accounts at the IdP and IDC.

The LoA levels and the way there are achieved are presented below:

- LoA 1 and LoA 2 can be achieved through password tokens or soft cryptographic tokens.
- If the soft cryptographic tokens are used for multi-factor authentication, then LoA 3 can be achieved (i.e. FIDO authentication).
- If hard tokens are used for multi-factor authentication, then LoA 4 can be achieved (i.e., FIDO authentication with keys stored in TEE).

The Account Management Module provides recovery mechanisms for cases when users cannot authenticate to their Identity Consolidator accounts. These mechanisms are represented by sets of questions defined at the registration with the Identity Consolidator, SMS verification and proof of possession of online accounts.

3.2.5 Identity Management Module

The Identity Management Module is a framework that offers users various functionalities in order to securely manage their identities in INCOGNITO. The module is divided into two submodules: The Identity Management and the Consent Management. It contains an identity matrix which contains different types and range of identifiers and unique user identity attributes. Among the functionalities the Identity Management offers, we can count the transfer of reputation and other identity attributes between Identity Providers, the creation of partial verifiable profiles which enable the user to share only selected attributes with a certain Service Provider or the consent for the management of their identity attributes.

The Identity Management Module enable two interfaces for the user:

- **Identity Data Management:** The user has access to its data stored in the Identity Repository and can update some attributes if the Identity Consolidator’s policies for those attributes allows this. The updating process can sometimes trigger the identity acquisition and verification process. When the attributes have been updated at the Identity Consolidator, all the online services storing the respective attributes are informed regarding the change.
- **Shared Identity Attributes:** The user can see all the identity attributes that are shared with online services and the respective online services.

3.2.5.1 Identity Profile Management Module

The Identity Profile Management submodule enables the user to use an interface in order to manage Identity Providers’ and Service Providers’ access to their identity attributes. The user can use this module to share identity attributes between different Identity and Service Providers, transfer that happens according to the policies defined within the identity consent management module. The attributes can also be transferred to or from the Identity Consolidator, process which takes place by means of federated identities and online identity acquisition module. The user can only view, delete and insert attributes that comply with certain attributes. 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. The user can also determine the risk of Identity or Service Providers by inferring attributes that were not revealed to them. These risks are calculated through statistical analysis of correlated identity attributes. As mentioned above, the user is also enabled to create and manage partially verifiable profiles.

3.2.5.2 Consent Management Module

The Consent Management Module enables the users and Identity providers to give their consent regarding the shared identity attributes. INCOGNITO automates this process through policies that are stored at the Authorization Server which will then be able to give a client access to some resources asynchronous to the presence of the user who owns the attributes.

The Consent Management Module has two submodules:

- *User-defined policy*: The user can define policies at the Authorization Server in order to control which attributes should be shared and with what Identity or Service Providers. Using these policies, the user is assured that his identity attributes are shared according to his consent without even being online at the moment of the user data sharing.
- *Identity Provider-defined policy*: This submodule is responsible for obtaining policies for individual attributes from Identity Providers, ensuring that the Identity consolidator reveals attributes to Service Providers according to these policies.

3.2.6 AI-based Assistant Backend

The INCOGNITO project aims to use an advanced UI/UX AI-based assistant (AIAS) which will focus on facilitating identity management tasks by offering an intuitive, user-friendly interface between the INCOGNITO services and the end-users. The AIAS will be present in both the mobile and the web version of the INCOGNITO user interface and will be able to inform the user about Identity management related tasks, but also to provide and execute actions on them. In order to achieve this, the basic architectural components will include two communication components, one residing on the client (web and mobile) and another one residing on the server (IDC). On the IDC there will be an engine that will perform correlations based on user data, user input or inquires and a set of rules that will be generated by machine learning algorithms as well as pre-defined actions to support the INCOGNITO functionalities.

For the implementation of the back-end machine learning component we will use Rasa HQ⁸. The advantages over commercial APIs provided by third party, are privacy and avoiding vendor lock-in, since it is not always possible to download all the data and logic from third party APIs.

On the client side (web and mobile) we will implement upon existing open-source chatbot UI implementations for Android and the Web (e.g.). The client-side of the AI-based Assistant will act as the interface that captures the user input and inquires and passes them to the client communication component that will in turn send them to the back-end for analysis. All the

⁸ <https://github.com/RasaHQ/rasa>

communication between the components residing in the client and the server will be done through REST-APIs.

The main goal of the AI-based Assistant is to guide the end-user through various aspects and tasks for managing his identity. The AI-based Assistant Backend will be integrated into the IDC, which will include all structures, scripts, and machine learning models. The machine learning models will be specifically trained and tailored to the intricacies and special requirements of a single, highly structured type of question-answering problem, to achieve high accuracy in the produced answers. In addition, the assistant will assist the user in preserving their privacy while accessing services on the web and, at the same time, will enable them to identify possible risks of de-anonymization. Moreover, the assistant will interact with the user through verbal cues and will be able to get feedback through web and mobile applications.

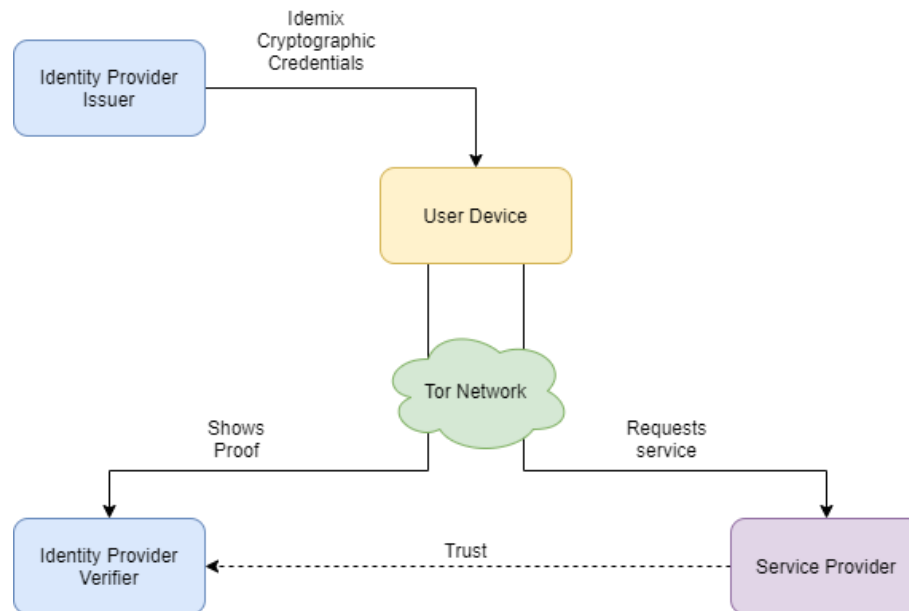
Rasa has two components, the NLU and the dialog tracker (called CORE). The NLU (natural language understanding) is in charge of understanding what the user writes, it first classifies the input sentence into a set of possible user's intents and also extracts the entities from the sentence, like names, numbers, locations, etc. That information is then passed to the CORE, which is in charge of keeping track of the conversation and deciding what action to take or what response to give to the user.

The NLU is trained with sample sentences annotating the intent and the entities, and the CORE is trained providing sample conversations called stories, so that the agent learns the correct correspondence between user input and the action to take. As an example, training an FAQ would be as simple as classifying the input as any of the possible questions and then providing simple sample conversations to the core composed of pairs of question-answer.

The training and deployment of both NLU/Core components require a reasonable amount of effort as they can expose a REST API, POSTing the user input as a sentence and having as a response all the necessary suggested actions that the user should take in JSON format. For both components there are several model architectures to train, with simple sklearn-based or with deep neural model like the state-of-the-art BERT[6] for the NLU component. At the back-end, a database will also be used to store all the related information and to enable the implementation of the AIAS in terms of the rules and the correlations needed. This database might be also implemented in the existing IDC database.

3.2.7 Tor Network

To be able to anonymize the user's IP address and all of the information associated with it, a Tor network library will be integrated in the User Device. The user will have the ability to access Service Providers by using specific identity attributes and verify his cryptographic credentials to the Identity Provider anonymously through the Tor Network. It will be an extra level of security by encapsulating the cryptographic technologies (Idemix), that's why in this scenario communication between the User Device and the Service Provider and the Identity Provider Verifier should go through Tor network.



3.2.8 Blockchain

The Identity Consolidator will also serve as a node at the blockchain network. The interactions between the IDC and the users regarding the creation/modification of identity attributes access control policies will be logged, as well as the corresponding communication with the SPs. Blockchain offers advantages like immutability, transparency to the network members and accountability. On top of that, the identity attributes are going to be stored on the blockchain, offering an easy and fast way to ensure that the information that has been communicated to the SP has not been tampered with or subjected to a man-in-the-middle type of attack. That way, disputes can be swiftly resolved or completely avoided, and security is enhanced ensuring privacy and smooth operation of the predefined processes.

3.2.9 Identity Repository and Third-Party API

The identity attribute values are stored in the Identity Repository, which essentially is a database. It resides on the IDC and the data obtained by the identity providers collaborating with the IDC is stored there. The identity providers may come in the form of user input, or online social media (e.g., Facebook, IEEE membership, etc.), which naturally may result in different data formats. The identity repository processes the data and homogenizes it in a default format. More specifically, the identity repository will store information in a categorized manner. A table will be used in order to log the user’s primary information, like name, age, etc. Which will be accompanied by a unique key assigned to each one of them. Secondary information will be stored in another table; this data will be correlated with the corresponding user by utilizing the unique key. Secondary information is defined as data related to contact information, media, positions, etc. and may have stemmed by identity providers or user input.

The following figure represents this model:

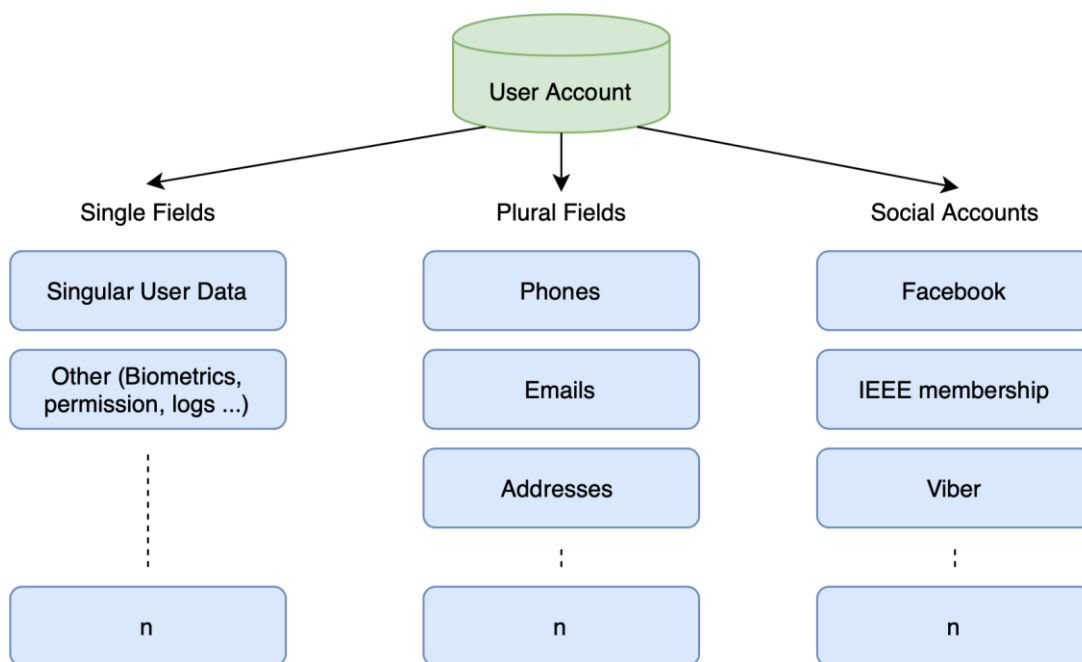


Figure 9. Storage Model

The IDC needs to be able to communicate with both the IdPs and the SPs in order to exchange information, along with the user device. In order to achieve that goal, we utilize a third-party API that makes it possible for the SPs to transfer information to and from the IDC, and the IDC can issue the required cryptographic credentials at the user device.

3.3 Identity Provider (IdP)

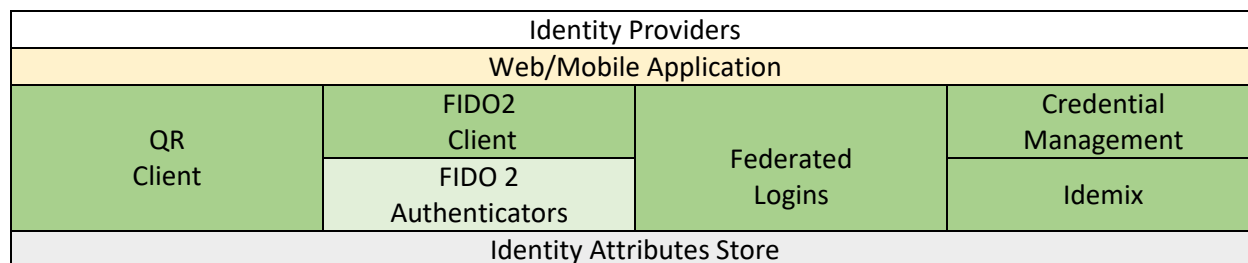


Figure 10. User’s Device Protocol Stack View

Since INCOGNITO is a platform whose main interest is user security and privacy, Identity Providers play a major role in authentication. An Identity Provider is a trusted system that authenticates users on behalf of another web resource, such as Service Providers. The functioning

principles relies on federated authentication which is an agreement between multiple organizations to use the same identification data to access different applications or services. In this case, an Identity Provider is an authenticating party where the users have their credentials for login. A third party (a Service Provider in INCOGNITO) trusts the authenticating party. Therefore, whenever a user wants to access the services provided by a Service Provider, it authenticates to an Identity Provider which then issues a token which will be used by the user to prove to a third party that he is who he claims to be.

Considering the leak of passwords and sensitive user information that took place in the last years as a result of data breaches and other security attacks that happened to the service providers' user credential databases, using an Identity Provider adds multiple layers of security such as multi-factor authentication or strong encryption. Using an Identity Provider also spares the user of creating and remembering multiple passwords and the service providers of storing and protecting user information.

In INCOGNITO, there are two modules that communicate with an Identity Provider, each with different roles but both use the same federated login approach to communication. The Service Provider makes use of the authentication information stored at the Identity Provider in order to avoid storing data about its users. Identity Providers holds identity attributes about its users and it also enables the cryptographic credential creation directly on the user's device through the credential management module which makes use of the identity attributes and follows the FiWARE specification which enables the Idemix cryptographic credential stack in charge of credential generation. The credential management module is also responsible for credential verification which results in trusted identity attributes which will be transferred to the Service Providers by means of the OpenID specification and to the Identity Consolidator to be stored and backed up in case of an Identity Provider failure.

The Identity Providers rely on two key servers. The first one – FIDO 2.0 server – represents the way through which users register and authenticate at the Identity Providers. The second one – QR authentication server – serves the purpose of allowing the user to access a service from another device or browser, if it is already using that service on their user device. A QR Client existent on the user's device will enable the user to scan a QR code and send it to the QR Authentication server for verification. In this way, the Identity Provider will be able to authenticate the user and give access to resources from a different device than the one he scans the QR Code from.

3.3.1 FIDO2 Server

The FIDO2 server is the component on which the IdP relies for registration and authentication. The Identity Provider has to deploy the FIDO2 server on their premises so that it can make use of its services whenever a user needs to access a Service Provider.

Before using the federated authentication enabled by the Identity Provider, a user has to be registered with it. It is now when the user has to register at the Identity Provider through the FIDO2 server's registration protocol. The registration takes place only once and the following interactions between a user and the Identity Provider is mediated by the FIDO2 server's authentication

protocol. This authentication enables the user to further access the FiWARE protocol stack, which is an abstraction layer above the Idemix protocol so that new cryptographic credentials would be issued on the user’s device and then would be transferred to the Identity Provider. The cryptographic credentials will get to the Service Provider in order to allow the user access to its services.

For the FIDO Server, the gateSAFE fulfills a double role. Firstly, it offers TLS encryption for a secure communication. Secondly, it grants a routing function for redirecting the communication to the corresponding server: the FIDO 2 Server or the Identity Provider. The routing takes place based on the URL that the two application servers are deployed on, as seen in the diagram below.

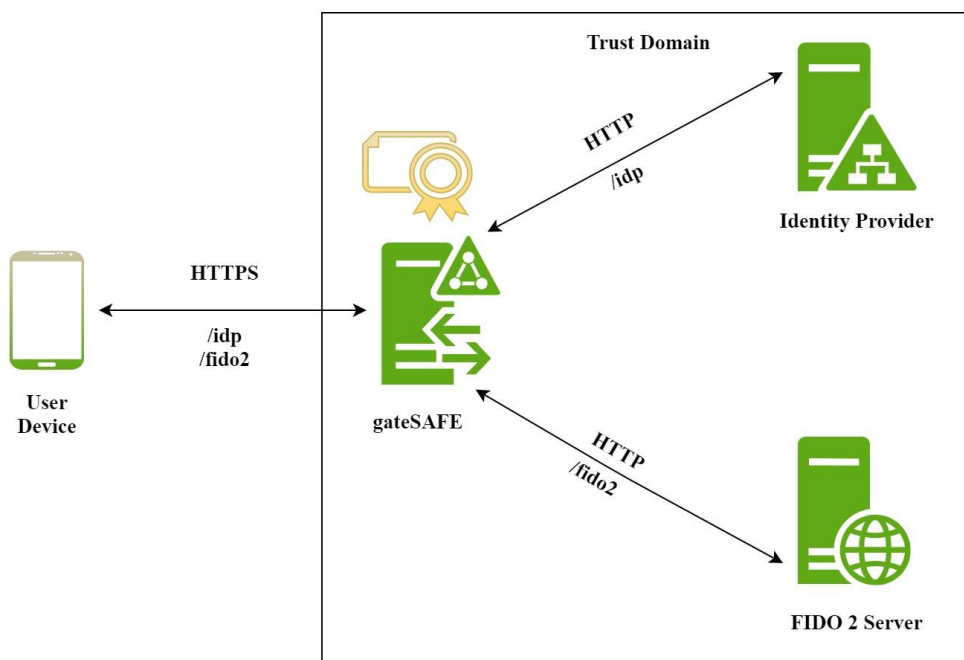


Figure 11. Routing through gateSAFE

3.3.2 QR Authentication Server

The QR Authentication Server enables the identity transfer from a mobile device to a desktop, therefore the user can access the protected resources at a Service Provider from multiple devices. QR Authentication works on the principle of federation, the Service Provider assigning the authentication responsibility to a QR Server which will complete the process according to a custom security policy. The authentication process starts when a user opens the portal from a web browser from a desktop device. At that point, the Service Provider generates a token associated with the current user session open at the user’s mobile device and sends an authentication request at the QR Authenticator module. The Service Provider request must contain a policy which will describe the security parameters according to which the QR Server will perform the authentication process. As

soon as the authentication request reached the QR Authenticator, the latter checks that the authentication request’s origin is reliable and generates an authentication message encoded in a QR code that will be displayed in the browser and scanned by the mobile application in order to allow the access to the running session from a desktop. The mobile application decodes the QR code in order to start an authentication session with the QR Authenticator. The application can authenticate at the QR Server by different protocols such as FIDO or TLS. The QR Authenticator verifies the authentication messages received from the mobile application and notifies the Service Provider about the status of the authentication. If the authentication is successful, the user starts an authenticated session from the browser and is redirected to the Service Provider. The steps of the process described above can be seen in the diagram below:

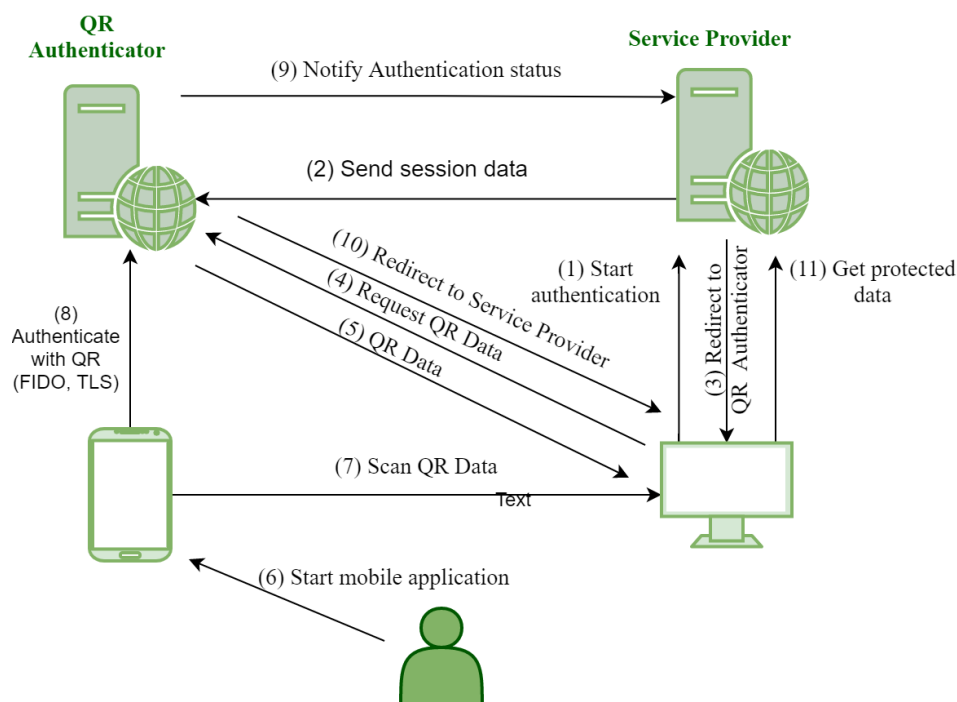


Figure 12. QR Authentication

Since the authentication can take place through the means of protocols as FIDO or TLS, the authentication message format is flexible and can be extended to be compatible with complex protocols as well. Each QR code generation carried out by the QR Authenticator is a unique one in order to avoid the duplicate authentication data. The generated code has a size limit so that the codes that are presented to the QR Device Client could be processed fast without causing QR code expiration. In order to enable the transmission of the QR code to take place safely, the communication happens through a TLS channel which binds the QR Server and the QR Web Client. Hence the QR Authenticator trusts the source of the QR code and allows the QR Device Client parse valid information and send authentication data to the QR Server. In order to avoid attacks, a TLS channel also binds the User Device with the QR server and the Web Client with the Service Provider.

3.4 Service Provider (SP)

Service Provider	
Web/ Mobile Application	
Federated Logins	
Access Control Policy Reasoning Tool	
Machine Learning and Data Mining Techniques (e.g., Risk Quantification)	Service Provider-specific Business Logic

Figure 13 Service Providers Protocol Stack View

Service Providers are already available in the market and used by the consumers. In order to achieve wide adaptation of the INCOGNITO platform by multiple Service Providers, we put effort into applying just essential changes on the SPs, that will be easier adopted with few modifications. SPs often have certain requirements that need to be met by the users in order to utilize their services. As defined in ReCRED, the XACML-based Access Control Policy Reasoning Tool is used by the Service Providers in order to define access control policies regarding the users’ identity attributes that are required in order to gain access to their services. These attributes are acquired from the Identity Consolidator, as users do not share directly information regarding their identity with the Service Providers. The OpenID Connect protocol combined with a cryptographic credentials stack, such as Idemix, are utilized for this purpose and deliver the attribute(s) needed by the Service provider in order to grant access to the user.

The INCOGNITO will also incorporate a blockchain solution in order to boost its privacy and security. The Service Providers will be part of that network, running an endpoint and participating at the submission of transactions on the blockchain, while they will be held accountable for their actions throughout the network’s lifecycle. More details about the blockchain integration are presented in Section 4.4.

4 Protocols and Technologies

4.1 FIDO 2

4.1.1 Overview

INCOGNITO integrates FIDO2 Protocol Specifications mainly because it offers a high protection against cyber-attacks (e.g., MITM, phishing) while also offering password less authentication for the users. This makes it a secure protocol and an easy-to-use solution, removing the overhead of memorizing passwords for all the accounts one has or for the data breach that could happen when using the same password for all them.

FIDO2 Protocol Specification is an upgrade to the already existing FIDO Specifications, namely FIDO UAF and FIDO U2F. In general guidelines, all three specifications of FIDO work on the same basic principle:

- When a user tries to authenticate with an online service, the latter one sends a request to the user device to ask for an attestation of authenticity.
- The user is prompted to choose and available authenticator that should be registered with the online service.
- The registration starts when the user unlocks the chosen authenticator through a user to device authentication such as entering a PIN or providing a fingerprint.
- Next, the authenticator generates a pair of asymmetric keys which is uniquely shared between the authenticator, user device and online service, therefore being associated to only one user account. The private key never leaves the authenticator, while the public key is sent to the online service where it will be stored in a database on the server side.

After the registration of the asymmetric key pair, every access to the user account will be allowed when a signed challenge will be verified at the server using the public key associated with the respective account.

Despite relying on the same functional principle, each one of the specifications has a different purpose.

FIDO UAF is oriented towards offering password less authentication. It relies on registering a user's device with an online service and it can offer password less access to the user's account only from the registered device. On this purpose, the user device has a FIDO UAF stack installed on it and it also has an embedded platform authenticator. This authenticator is registered with the online service and whenever the user wants to access his account, he will unlock the authenticator through a gesture on the user device and the authentication process will be verified at the server side.

FIDO U2F is oriented towards increasing security through a second-factor authentication. While the old authentication method with username and/or password is still available, the online service will also request the user to present a second factor device such as a FIDO Security Key. Although a password is still involved, it can be reduced to a 4-digit PIN since the second factor device has the function to deliver a stronger authentication method. The registration though, involves registering the roaming authenticator since the beginning just as it is done in the FIDO UAF specification, namely by unlocking the second factor device in order to register the public key with the online service.

FIDO2 is up to a point, a unification between FIDO UAF and FIDO U2F, but it also brings a new feature. FIDO2 authentication supports password-less authentication, second-factor, but it also increases security by adding multi-factor authentication on top of these authentication methods. FIDO2's main importance is that it allows a user to authenticate to online services on mobile devices, as well as on desktop environments. Registering multiple authenticators with a single account, as much as adding FIDO Authentication support inside web browsers through a standard web API known as WebAuthn have made this feature possible. Since it has been said that FIDO2 comprises both FIDO UAF and U2F, it is worth mentioning that it supports platform authenticators and roaming authenticators at the same time. This is a very useful feature when trying to recover

someone’s account since one can register with the online service mobile devices, as well as FIDO Security Keys or wearables.

So that all the new features brought by FIDO2 may be supported, it is necessary to split it into two main components:

- WebAuthn Standard which is responsible for the web authentication at the server side and is mainly useful for using multiple devices with a single account.
- CTAP2 Standard which is the new replacement for the ASM (Authenticator Specific Module) used in FIDO UAF in order to allow communication between the FIDO Client and the platform authenticator. Since FIDO2 uses roaming authenticators as well, it is necessary to define a standard of communication between the FIDO Client present on the user device and the authenticator present on a different device.

4.1.2 WebAuthn

WebAuthn is a standard that was developed by FIDO⁹ Alliance in collaboration with World Wide Web Consortium (W3C)¹⁰ and was declared an official web standard in March 2019. It is the main piece in FIDO2, offering users the ability to login to web applications using passwordless authentication.

It introduces Web Authentication API as a browser built-in piece of code which adds FIDO support to them. The standard revolves around the way the Authenticator, Client and Relying Party interact with this web API, while conforming with the security and privacy considerations imposed by the WebAuthn Specification.

The flow of authenticating users through web applications starts with a request sent by the Relying Party web application to the authenticator in order to create one or more public key credentials that should be registered with the account. The request is actually a call to the Web Authentication API which, with the help of the user agent, will make the request reach the client device where it will be either directed to a platform authenticator or a roaming authenticator using CTAP2 Standard. The authenticator sends an attestation to the requesting Relying Party, offering cryptographic proof of their properties, though not before they are unlocked by a user action such as fingerprint or PIN.

4.1.2.1 Web Authentication API

Web Authentication API is a script that offers a way of creating and using public key credentials with a browser. It sends requests to the authenticator and receives the responses with the help of the user agent on the client platform, but it never actually accesses the information since it is sent and received as objects.

The security of the communication that takes places through this API is maintained by the authenticator and the user client working together. The authenticator always makes sure that the

⁹ <https://fidoalliance.org/how-fido-works/>

¹⁰ <https://www.w3.org/TR/webauthn/>

credentials are sent to the origin that requested them by signing and integrating them in all the responses, be it the attestation object created at the registration moment, or any other assertion objects created whenever the user authenticates. Regarding user privacy, the authenticator makes sure that each created credential is associated with a certain Relying Party ID (RP ID). Whenever it sends a response, it first has to receive from the client the requesting RP ID in order to check there are actual credentials that match the requester and then send the response to the Relying Party. In this way, the authenticator makes sure that credentials are only used by the Relying Party that requested their creation without disclosing to a different Relying Party, other existent user accounts and their origins.

There are two main interfaces in Web Authentication API serving the creation of public key credentials, as well as their transport to the requesting Relying Party:

- *PublicKeyCredential* Interface which inherits from *Credential* interface and contains the attributes that must be returned to the caller during registration and authentication to a Relying Party.
- *AuthenticatorResponse* Interface which is used by the authenticator to response to a Relying Party request. It contains a JSON serialized data received from the client which receives it, for its part, from a Relying Party sender.

4.1.2.2 WebAuthn Authenticator Model

WebAuthn Authenticator Model describes and abstract model of interaction of the Web Authentication API with the authenticator. It is not an already made implementation. This is something each client platform implements in its own way, but the implementation should offer the same interaction and result when used with the Web Authentication API.

The WebAuthn Authenticator Model describes data encoding and logical operations exposed to the client and the Relying Party. It does not impose a certain protocol of communication between a roaming authenticator and a client device such as USB or NFC.

Relying Parties can choose the type of the authenticator to register with through calling certain methods from Web Authentication API, which will be propagated to the methods described by the WebAuthn Authenticator model and will be sent in the end to the authenticator itself. The authenticator's main function is to provide signatures for the WebAuthn. The two purposes for which a signature is needed are:

- To create an **attestation signature** which is needed to prove to a Relying Party the properties of the signing authenticator. It is produced when a registration takes places, therefore when a new public key credential is created.
- To create an **assertion signature** which is needed to prove to a Relying Party that the user who initially allowed the creation of a public key credential is the one who also accepted the authentication or transaction in progress for which the assertion is being created.

The main abstract operations to interact with an authenticator are:

- *authenticatorMakeCredential* operation invoked when a new public key credential request is sent from the Relying Party.

- *authenticatorGetAssertion* operation invoked when an authentication or transaction is taking place. The Relying Party sends a request to the authenticator to get an assertion in order to be able to allow the user’s access to the account.
- *authenticatorCancel* operation invoked by the client in an authenticator session. It has the effect of terminating any *authenticatorMakeCredential* or *authenticatorGetAssertion* operation currently in progress in that authenticator session.

4.1.2.3 WebAuthn Relying Party Operation

The Relying Party is the entity that takes care of creating objects that will be sent to an authenticator in order to register or authenticate a user. Upon registration, the Relying Party’s script receives a *PublicKeyCredential* object containing an *AuthenticatorAttestationResponse* from the authenticator. The content of these objects is delivered to the Relying Party’s server where the public key credential is stored and associated with a user’s account.

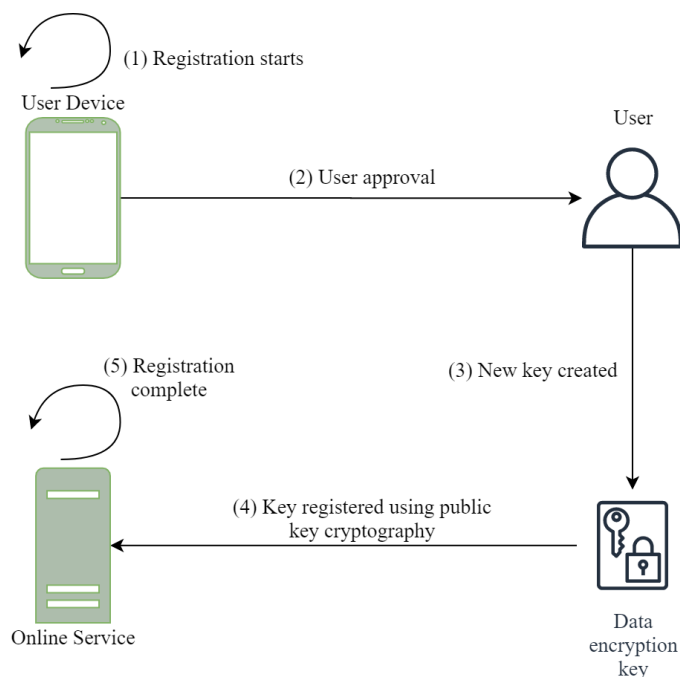


Figure 14. FIDO Registration

Upon authentication, the Relying Party’s script receives a *PublicKeyCredential* object containing an *AuthenticatorAssertionResponse* from the authenticator. The content of these objects is delivered to the Relying Party’s server where the signed challenged received from the authenticator is verified with the public key credential stored at the Relying Party's server.

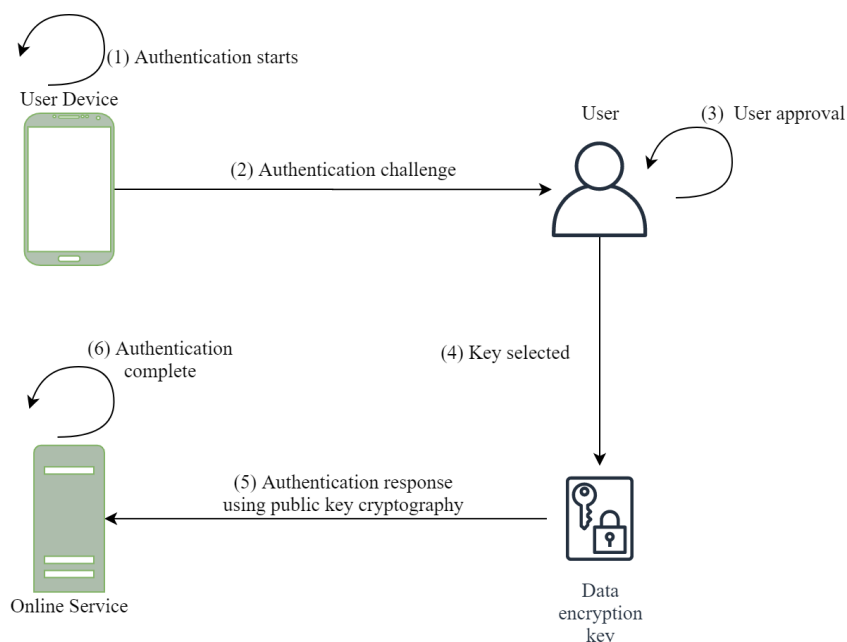


Figure 15. FIDO Authentication

4.1.3 CTAP2

Client to Authenticator Protocol 2 (CTAP2)¹¹ is an application layer protocol meant to create a communication between a roaming authenticator and client platform. The protocol works according to the following steps:

- The client platform creates a connection with a roaming authenticator.
- The client platform calls the *authenticatorGetInfo* command to obtain information about the capabilities of the authenticator.
- The client platform sends a command to the authenticator if it is able to support it.
- The authenticator replies with the requested data or an error.

CTAP2 Protocol is composed of three main parts:

- **Authenticator API** which is an abstract API that describes the way a client platform interacts with the authenticator. The operations are just like API calls, accepting parameters and returning messages as output or errors. The main operations are similar to the ones described before in the WebAuthn Authenticator Model, since they convey the same data that has to reach the authenticator, just being discussed from another entity's perspective.
- **Message Encoding** which is the encoding format used by a client platform before calling an API method. The authenticator will respond with a message encoding in the same format. The encoding used is CTAP2 canonical CBOR which represents a light-weight

¹¹ <https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-client-to-authenticator-protocol-v2.0-ps-20190130.html>

form of encoding. It is necessary to reduce the complexity of the messages since transport mediums can be bandwidth-constrained such as BLE. The JSON serialization used in the previous FIDO Specification is too heavy-weight for the new way of transporting messages to and from roaming authenticators.

- **Transport-specific Binding** which is the format of communication specific to a certain transport protocol such as USB, NFC, BLE used to convey information between the roaming authenticator and the client platform.

4.1.4 FIDO2 integration with INCOGNITO Architecture

INCOGNITO integrates FIDO2 Specification in its protocol stack, along with other specifications such as Privacy Attribute Based Access Control in order to allow the user a passwordless, multi-device authentication while keeping its privacy through untraceability and unlinkability.

The components of the FIDO2 stack are distributed among every part taking part in the authentication: on the online services side, on the user device, as well as on other devices able to act as an authenticator such as wearables.

1. FIDO2 Server is implemented in both Identity Consolidator and Identity Provider. It is responsible for:
 - interacting with the Web Authentication API script supported by the browsers in order to send request to the client on the user device.
 - verifying the validity of the FIDO2 messages received from the client.
 - managing user accounts and their associations with the public key credentials created by the authenticator
2. FIDO2 Client runs on the user device and is in charge of the following operations:
 - communicating FIDO2 messages to and from the Relying Party. It interfaces with a user agent that can be a browser plugin which should implement the specific Web Authentication API specific to FIDO2 Protocol. The FIDO2 Server running at the Relying Party sends a message to the web application running at the online service which will redirect the message to the user agent running on the user's device which will forward next the messages to the user client.
 - communicating with the authenticators through CTAP2 protocol which sends messages encoded in CTAP2 canonical CBOR format.
3. Authenticator can be a platform authenticator (embedded on the user device) or a roaming authenticator. Its functions are to:
 - communicate with the client through CTAP2 protocol and interfaces with the user device through different transport protocols such as USB, NFC, BLE.
 - generate a pair of asymmetric keys which will be used to associate the authenticator with the user's account at the Relying Party. In order to maintain the security of the generated

keys, the authenticator should run the key generation process inside a trusted execution environment (TEE).

4.2 OAuth 2.0

OAuth 2.0¹² is considered to be the most widely used and adapted protocol upon which authentication solutions are implemented. It offers interoperability among any kind of devices (e.g. desktop applications, smartphones, IoT devices etc.) by enabling HTTP access to be granted to applications that need corresponding resources owned by a different entity. Both OpenID connect and UMA technologies that we are going to utilize in order to implement the INCOGNITO platform are based on OAuth 2.0; the authorization layer provided clearly differentiates clients from resource owners. In order for a client to access resources that belong to a different owner and reside on a resource server, new credentials are issued – usually an authentication token – for the client with the owner’s consent. This token is utilized by the client in order to access the resources that reside on the resource server. Below we will analyze how this concept applies specifically for the OpenID Connect and UMA solutions.

4.2.1 OpenID Connect

OIDC is defined as an identity layer that has been implemented on top of the OAuth 2.0 protocol. It makes it possible for clients to verify the corresponding identity of a user by utilizing the authentication procedure that takes place on an authorization server. In essence, OIDC concerns the user's authentication compared to OAuth 2.0 which deals with resource access and sharing. When utilizing OIDC protocol, there are three options that the authentication procedure may follow, their main difference being the way the ID and Access token are passed to the client:

- Authorization code flow: This flow is widely used by common web applications and programs executed on smartphones and IoT devices. It redirects the user to a login page where authentication and consent procedures take place. After that step, the ID and Access tokens are issued.
- Implicit flow: Applications that are executed on the browser are primarily utilizing this flow, as the absence of a backend requires that an ID token is sent to the client without the procedure described at the authorization code flow taking place.

As it is mentioned before, the OIDC reuses the OAuth 2.0 protocol and parameters and extends it to introduce an Identity Layer through the following additions:

- Along with the access token, an ID token is returned, which is basically a JSON Web Token with identity claims (user information).
- An endpoint regarding user information is used, which returns identity attributes that correspond to an access token.

4.2.1.1 Authorization Code Flow

Web applications are executed on servers that provide backend operations. The Authorization Code Flow is used in that case, which essentially means that the application trades an Authorization Code for an Access Token.

¹² <https://oauth.net/2/>

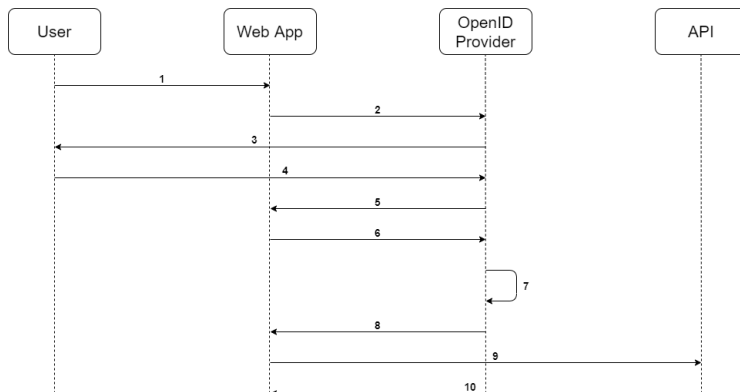


Figure 16. Authorization code flow

The above figure illustrates the Authorization Code Flow and the whole process is described below:

1. The user tries to login at the web application.
2. The web application passes on the request to the OpenID Provider.
3. The OpenID Provider redirects the user to the corresponding Authorization Server to login and give the corresponding consent.
4. The user utilizes his/her credentials to authenticate and grant permissions to the web application.
5. The OpenID Provider issues a one-time use authorization code.
6. This token is processed by the OpenID Provider’s corresponding endpoint along with the application's Client ID.
7. The OpenID Authorization Server performs a verification regarding the authorization code and Client ID.
8. The OpenID Authorization Server sends back an ID and an Access Token.
9. Now the web application can utilize the Access Token to communicate with an API to access user identity attributes.
10. The API sends back the requested attributes.

The exchange of the above messages is done with HTTPS or HTTP protocol, with GET and POST methods.

4.2.1.2 Implicit Flow

Instead of the Authorization Code Flow, OpenID Connect can be realized through the Implicit Flow which is mainly addressed to Public Clients. This flow does not make use of an Authorization Code, just an ID Token that is traded at the corresponding endpoint. This flow is not recommended for big scale deployments, but it is very useful in cases where the application requires just the ID Token to authenticate the user.

The figure below shows the Implicit flow procedure.

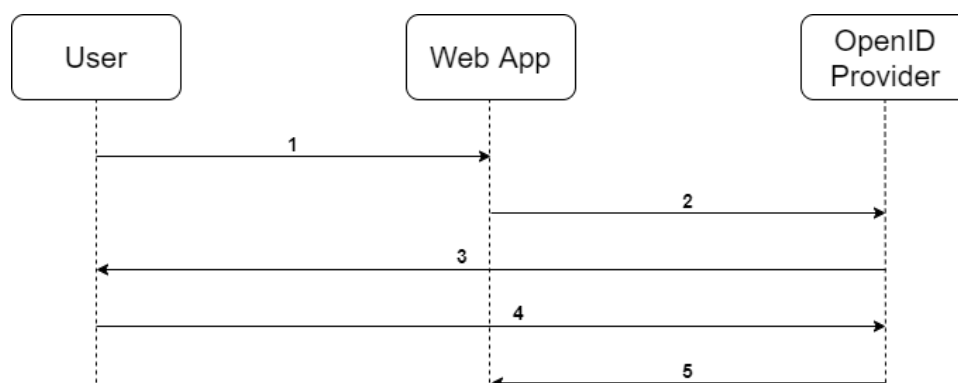


Figure 17. Implicit flow

1. The user attempts to login at the web application.
2. The user is guided to the OpenID Authorization Server, providing a “response_type” parameter that indicates the ID token’s format that the user requires.
3. The OpenID Provider redirects the user to the corresponding Authorization Server to login and give the corresponding consent.
4. The user utilizes his/her credentials to authenticate and grant permissions to the web application.
5. The user is provided with an ID Token.

The above requests and responses are made in the same manner as with the Authorization Code Flow. The exchange of the above messages is done with HTTPS or HTTP protocol, with GET and POST methods.

4.2.2 User-Managed Access (UMA)

User-managed access (UMA) is a federated authorization standard protocol built on top of OAuth. As described by the creators, the purpose of the protocol specifications is to “enable a resource owner to control the authorization of data sharing and other protected-resource access made between online services on the owner’s behalf or with the owner’s authorization by an autonomous requesting party”.

UMA defines 2 sets of specifications:

- UMA 2.0 Grant for OAuth 2.0 Authorization: specifies how a client should use a permission ticket to request OAuth access token in order to gain access to a protected resource.
- Federated Authorization for UMA 2.0: defines a means for a UMA-enabled authorization server and resource server to be federated in a secure and authorized resource owner context.

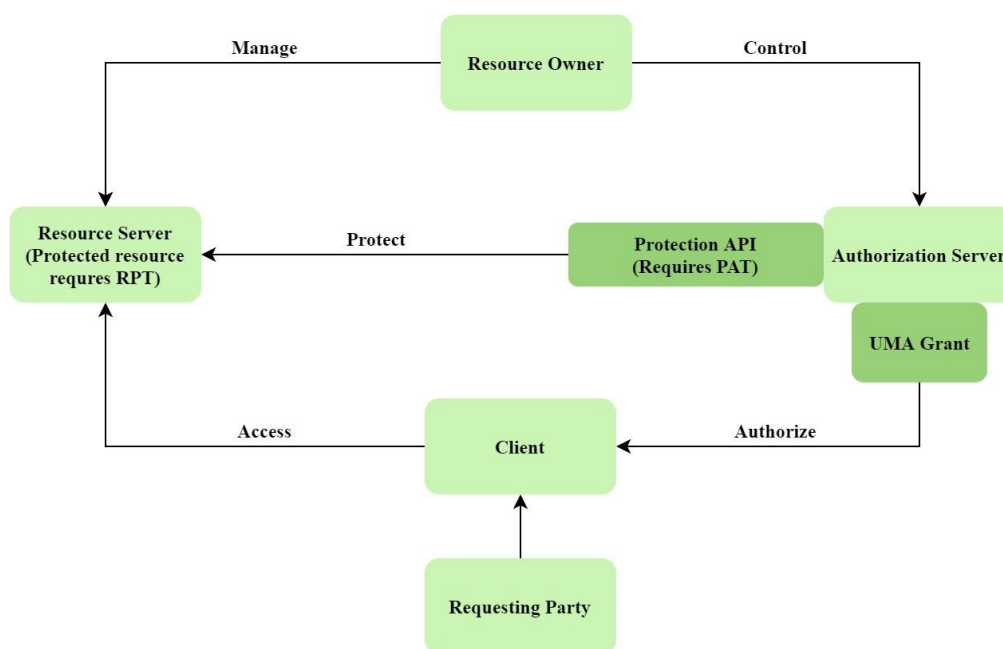


Figure 18. UMA general flow

UMA extends the definitions of entities defined by the OAuth protocol in order to accommodate a new role, the requesting party. As a result, the following roles are defined:

- Resource owner: an entity which can grant access to the protected resources.
- Requesting party: an entity, natural or legal person, which seeks access to a protected resource. This entity may or may not be the same party as the resource owner.
- Client: an entity capable of requesting access to protected resources with the acceptance of the resource owner, in the name of the requesting party.
- Resource server: an entity capable of granting access to protected resources in the resource owner’s behalf
- Authorization server: an entity that protects and grants access to the protected resources hosted on the authorization server on the resource owner’s behalf

4.2.2.1 UMA Grant

The UMA grant¹³ defines an extension OAuth 2.0 grant that enhances OAuth capabilities in the following ways:

- The resource owner authorizes protected resource access to clients used by entities that are in a *requesting party* role. This enables party-to-party authorization, rather than authorization of application access alone.
- The authorization server and resource server interact with the client and requesting party in a way that is *asynchronous* with respect to resource owner interactions. This lets a

¹³ <https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-grant-2.0.html>

resource owner configure an authorization server with authorization grant rules (policy conditions) at will, rather than authorizing access token issuance synchronously just after authenticating.

The authorization flow of the UMA Grant starts with an initial resource access request made by the client to the resource server with an invalid or inexistent access token. The resource server gets on behalf of the client, a permission ticket from the authorization server. This will serve the client to access the authorization server in order to obtain a Requesting Party Token (RPT) to access the resources it needs. RPT is an OAuth access token whose issuance takes place as a consequence of claims gathering and an authorization assessment at the authorization server. The authorization server has a predefined endpoint where a client can find a discovery document mentioned in the UMA specification as Authentication Server Metadata. It is here where the client can find all the endpoints and other kind of metadata exposed by the server.

4.2.2.2 Federated Authorization

As mentioned in the Overview section, the Federated Authorization¹⁴ is focusing on the interaction between the resource server and the authorization server. This communication is mediated by a Protection API exposed by the authorization server which provides the following three endpoints:

1. Resource Registration Endpoint
2. Permission Endpoint
3. Token Introspection Endpoint

Before interacting with the Protection API, a resource server needs to obtain OAuth client credentials from the authorization server. These credentials come as an access token – namely Protection API Access Token (PAT).

1. Resource Registration Endpoint

Since a resource server has obtained a PAT, it is now allowed to put resources under the protection of the authorization server. The resources are registered through the Resource Registration Endpoint which supports five registration options:

- Create resource description - registers a new resource to the authorization server using the POST method.
- Read resource description - reads a previously registered resource description using the GET method.
- Update resource description - updates a registered resource description, by completely replacing the previous resource description, using the PUT method.
- Delete resource description - deregisters a previously registered resource description using the DELETE method.
- List resource descriptions - lists all previously registered resource identifiers for this resource owner using the GET method.

¹⁴ <https://docs.kantarinitiative.org/uma/wg/rec-oauth-uma-federated-authz-2.0.html>

2. Permission Endpoint

The Permission Endpoint is where the resource server accesses the authorization server in order to get a permission on behalf of a client which requests access to a resource. When the client reaches the resource server without a RPT, the resource server obtains a permission for the client which will be used by the latter to get a RPT from the authorization server.

3. Token Introspection Endpoint

The Token Introspection Endpoint is a checkpoint for the authenticity and validity of a Requesting Party Token. When a client finally obtains a RPT and presents it to the resource server in order to get access to a resource, first the resource server checks if the RPT is valid at the Token Introspection Endpoint and then the flow of UMA finally ends in an access or a denied permission to a resource.

4.2.2.3 *UMA Flow Diagram*

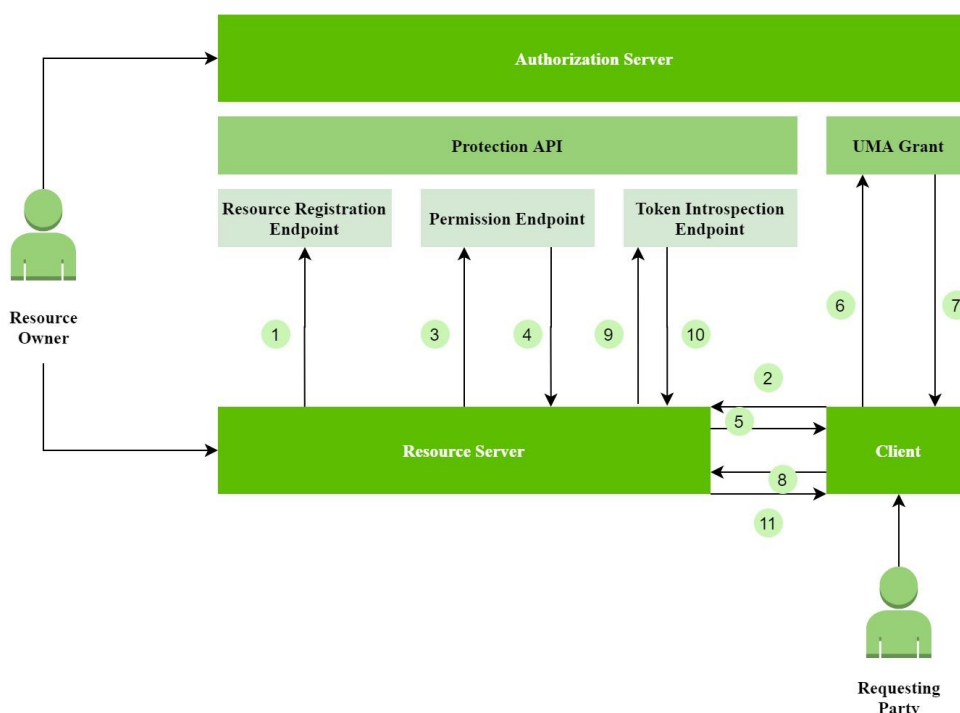


Figure 19. UMA Flow

The steps shown in the above diagram form the whole UMA flow, from the resource request to the resource access and are detailed in the following lines:

1. The resource server puts the resources under the authorization server’s protection.
2. The client acting on behalf of a requesting party, makes an access request to the protected resource without an RPT or with an invalid one.

3. The resource server requests permissions (resource identifiers and scopes associated with the requested resource) from the authorization server.
4. The authorization server answers with a permission ticket.
5. The resource server returns to the client the URI of the authorization server and permission ticket with which it can make a request for a RPT.
6. The client makes a request for an access token at the authorization server using the permission ticket and additional claims.
7. After an assessment algorithm, the authorization server issues a RPT.
8. The client makes a resource request, but this time with a valid RPT.
9. The resource server introspects the RPT at the authorization server in order to assess its validity.
10. The authorization server returns a token introspection response.
11. If the response is positive, the resource server gives the client access to the requested resource.

4.2.2.4 UMA Implementations

User Managed Access is a project of the global consortium Kantara Initiative whose mission is to bridge the identity community with actions that protect users' privacy and security in order to create more trustworthy networks. On their official website, the consortium lists all the known UMA implementations. After a thorough research on every project found in there, three main options stood out due mostly to their open source and complex nature. They are presented in the following three subdivisions.

1. Gluu

Gluu¹⁵ Server is a project run by the Gluu Federation and it is an open-source software (FOSS) solution for identity and access management (IAM). It is a container distribution which incorporates different open-source projects that together come to create a complex IAM solution that covers different use cases such as:

- Single sign-on (SSO)
- Mobile authentication
- API access management
- Two-factor authentication (2FA)
- Customer identity and access management (CIAM)
- Identity federation

These use cases come to be realized through the implementations of different web standards for authentication, authorization, federated identity and identity management:

¹⁵ <https://gluu.org/docs/gluu-server/>

- OAuth 2.0
- OpenID Connect
- User Managed Access 2.0 (UMA)
- SAML 2.0
- System for Cross-domain Identity Management (SCIM)
- FIDO Universal 2nd Factor (U2F)
- FIDO 2.0 / WebAuthn
- Lightweight Directory Access Protocol (LDAP)
- Remote Authentication Dial-In User Service (RADIUS)

Gluu Server is supported on multiple operation systems such as Docker, Ubuntu 16 and 18, CentOS 7.x but it is not supported on Windows.

Despite the fact that Gluu is a complex identity and management solution, for the INCOGNITO project it is of interest only the User Managed Access implementation. There are three main components that create the whole UMA flow:

- Gluu Server
- Gluu Gateway

Gluu Server acts as an authorization server for UMA. It implements the two main components: UMA Grant and Federated Authorization by exposing endpoints, keeping policies for authorization and implementing an assessment algorithm in order to decide whether a client is or is not suitable for accessing a certain resource. But in order for this whole authorization flow to happen, it is first necessary to bridge the resource servers, authorization server and the clients together through authentication. Gluu server offers the authorization, but in order to be able to start this flow, a client needs first to be authenticated at the server as well as a resource server needs to be authenticated in order to put its resources under the authorization server's protection. And here is where Gluu brings into the equation another module – namely Gluu Gateway. It offers the possibility to authenticate through UMA bearer tokens or through OAuth 2.0 access tokens. This feature is offered by the Gluu Gateway plugins and by the oxd Server which represents a middleware that enables communication between the authorization server and a client.

2. WSO2

WSO2¹⁶ Identity Server is fully open source and is released under Apache Software License Version 2.0. WSO2 Identity Server provides, just as Gluu, an identity and access management solution. It is built on top of WSO2 Carbon which is a middleware platform whose main role is to allow developers to easily manage business processes and develop services. The Identity Server offers the users to deploy it on premise servers, private cloud or public cloud without configuration changes.

The WSO2 Identity Server is supported on multiple operating systems such as Windows, Ubuntu, Docker. It offers the following list of services:

- User stores and directories
- Authentication of users
- Authorization of users
- Single sign-on
- Provisioning
- Access delegation
- Password reset
- Self-registration
- Account locking

WSO2 Identity Server supports UMA 2.0 protocol and acts as the authorization server, enabling resources management and access control to them. The flow of UMA in WSO2 can be seen in the image below:

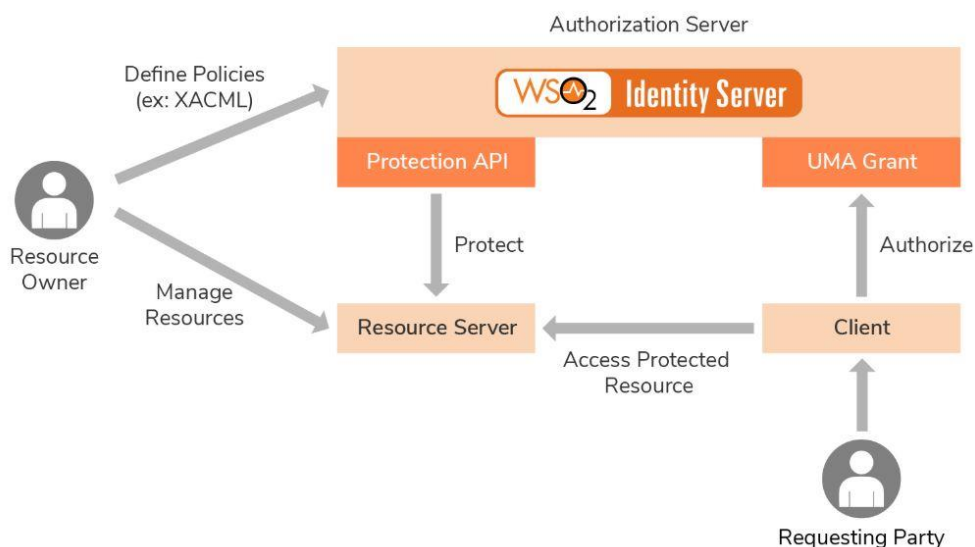


Figure 20. UMA Architecture in WSO2 Server

¹⁶ <https://is.docs.wso2.com/en/latest/>

3. KeyCloak

KeyCloak¹⁷ is a single sign on solution for web apps and RESTful web services led by RedHat. It is supported on Docker, Kubernetes, but it also comes as a standalone project ready to be run on Windows or Linux operating systems. It runs as a different server on your network and applications are secured by it. A browser application redirects a user from the application to the KeyCloak authentication server where the user enters his credentials. In this way, the user is isolated from the application and the application does not have access to the user’s credentials.

KeyCloak source code is bundled with the WildFly application server, but it also offers client adapters for other web servers such as Tomcat or Jetty. In order to manage different groups of applications and users, KeyCloak introduces the notion of realm which is equivalent of a tenant. There is a default realm called master which is meant to manage the KeyCloak. For any other application, a different realm should be created.

Some of the features introduced by KeyCloak are mentioned below:

- Single-Sign On and Single-Sign Out for browser applications
- OpenID Connect support
- OAuth 2.0 support
- SAML support
- Identity Brokering - Authenticate with external OpenID Connect or SAML Identity Providers
- Social Login - Enable login with Google, GitHub, Facebook, Twitter, and other social networks
- User Federation - Sync users from LDAP and Active Directory servers
- Kerberos bridge - Automatically authenticate users that are logged-in to a Kerberos server
- Admin Console for central management of users, roles, role mappings, clients and configuration
- Account Management console that allows users to centrally manage their account

4. Gluu vs. WSO2 vs. Keycloak

Since all the solutions mentioned above are complex, in order to take a decision for the most suitable alternative for the INCOGNITO project, a list of requirements was drew up. In this way, a parallel between the three UMA implementations would create a big picture of what each solution can offer.

¹⁷ <https://www.keycloak.org/documentation>

	Gluu	WSO2	Keycloak
Installation Configuration /	Complex setup and it requires many resources	Comes as an executable ready to install	Comes as an executable ready to install
User Interface	Graphic User Interface but it is a bit unclear	Graphic User Interface easy to use and understand	Graphic User Interface easy to use and understand
Supported Platforms	Docker, Linux Not supported on Windows	Docker, Linux, Windows	Docker, Linux, Windows
Architecture	Comes in many modules that can be installed separately	One executable	One executable
Middleware	Apache HTTPD / Jetty	Runs on its own WSO2 Carbon middleware which is based on Tomcat	Bundled with WildFly/ JBoss Application Server but it also offers client adapters for Tomcat / Jetty and more and it can be configured to run on each of them, but the documentation does not offer support in this direction
OpenJDK support	No	Yes	Yes
Admin UI	Yes	Yes	Yes
Open source	Yes	Yes, but it is not clear if the latest security patches are included in the binaries on their site. It can be packaged from the source code, but it is still unclear if the updates are public or not.	Yes
Development activity	Active development	Active development, new release in 2020	Active development, new release in 2020, often pull requests that get accepted

Documentation	Detailed. Very few details in the developer Guide.	Detailed. Very few details in the Developer Guide.	Very detailed, there are many action described step by step such as adding certificates, adding policies, securing apps etc. Detailed Developer Guide that approaches many scenarios The server source code is explained in Java docs.
Quick starts	No	Yes, but is seems to come as binaries, there is no access to the implementation. All the integration with the authorization server is described from the point of view of the web interface. There is no quick start that describes UMA flow.	It offers many quick starts whose source code can be found on Github. These are small applications that can be integrated with KeyCloak. They offer a small application that follows UMA flow. The source code offers a way to understand how to integrate applications with the server and how to create and deploy policies at the authorization server.

Table 1. Comparison between open-source Identity Access and Consent Management solutions

Considering all the details presented in the above table, Keycloak seems to be the best solution to suit the needs of the INCOGNITO project. But there are also a few aspects to be discussed. Since KeyCloak is bundled with WildFly Application Server which is mainly the new name of the old JBoss, it is necessary to mention that WildFly is a project in active development. There is an active community for developers, forums and new releases – the last version was in May 2020. Creating policies is based upon a very simple format and deploying them on the server is made through a simple command from the CLI. As a downside of Keycloak, we can mention the fact that it does not implement the Interactive Claims Gathering from the UMA Grant Specification. This endpoint is though mentioned to be optional and it is not a necessity regarding the way INCOGNITO is designed.

4.3 Privacy Attribute-based Access Control (PABAC)

One of the main goals of INCOGNITO is to provide a Privacy-Preserving Attribute-based Access Control (PABAC) solution through INCOGNITO’s architectural components on top of Open ID Connect. PABAC enables Service Providers that are not aware of cryptographic credentials to allow end-users to use already issued cryptographic credentials from various Identity Providers to get access to their service.

To achieve this, we need a cryptographic credential issuing Identity Provider (IdP) and a verifying IdP. Users can request the issuance of a cryptographic credential for one or more identity attributes from the issuing IdP, which can be the Identity Consolidator (IDC) itself that runs the cryptographic credential issuer stack. The verifying IdP acts as an Idemix verifier able to verify cryptographic credentials while the Service Providers (SPs) continue to run the OpenID Connect protocol. In other words, the user requests the issuance of a cryptographic credential of one or more user identity attributes from an issuing IdP (e.g., student and over 18). When the user tries to authenticate to an SP, he triggers a session with a cryptographic credential verifying IdP in order for that Verifying IdP to verify the validity of the cryptographic credential. Subsequently, the verifying IdP assures the SP that the user is a holder of a credential that proves that she is a student and is over 18 years old. After this seamless to the user authentication procedure, the user can have access to the SP's service/resources, knowing that his anonymity is preserved and no more than the needed identity attributes have been revealed to that SP. Additionally, federated PABAC offers two concepts of anonymity, namely untraceability and unlinkability. This means that no SP or IdP can track or link any credentials to the user or vice versa. Figure presents an overview of the PABAC components and how the cryptographic credentials verification is achieved.

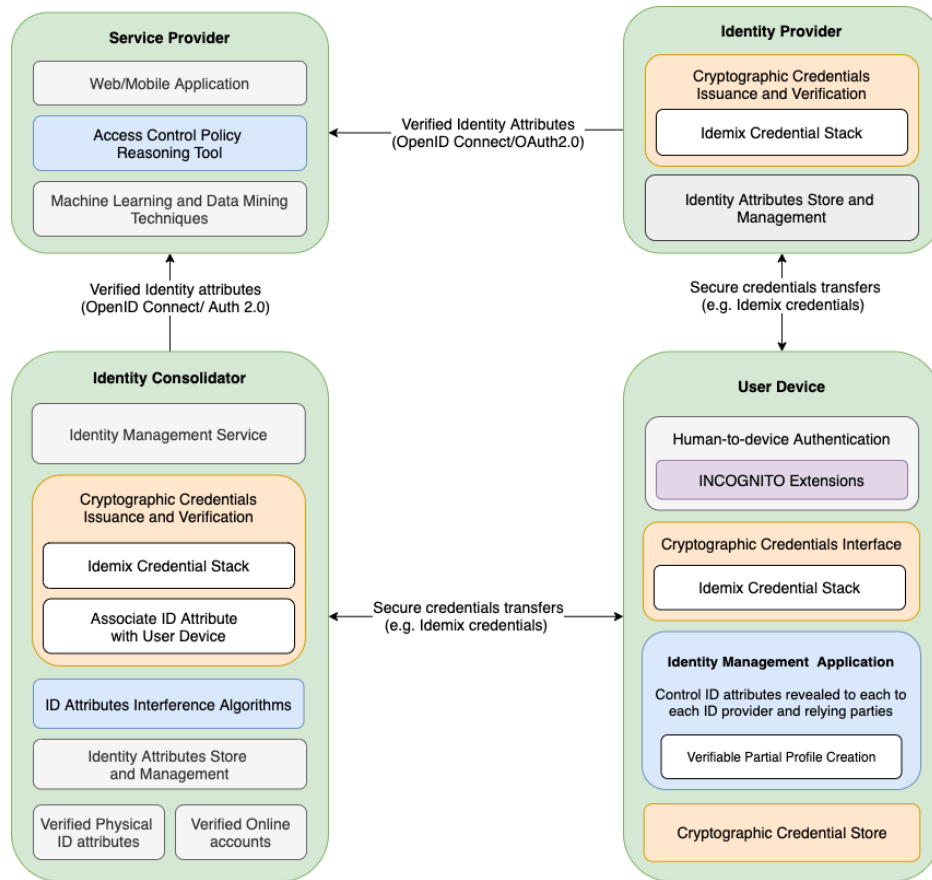


Figure 21. Attributed-Based Access-Control component view

4.3.1 Idemix

Internet transactions put at risk user’s privacy and security; therefore, it is necessary to include in INCOGNITO an additional protection layer. For this purpose, an anonymous credential system is integrated in this platform in order to allow users to access online services without disclosing their identity. Idemix (short for identity mix) is an identity management system based on anonymous credentials that was developed by [8]. A credential system is a system in which users can retrieve credentials from organizations and prove ownership of these credentials. Such a system is anonymous when transactions carried out by the same user cannot be linked; thus, providing privacy for users. Idemix is based on assigning a user multiple pseudonyms. Each time a user makes a transaction at a service provider, he has to prove he owns the credential associated with that respective pseudonym without actually revealing who he really is. There are four roles that define the flow of the Idemix protocol:

- **Issuer:** In INCOGNITO, the authority in charge of issuing credentials is played by the Identity Consolidator.

- **Recipient:** This role is assigned to the user device. After generation the credentials, the Identity Consolidator sends them to the user device where they are stored for future transactions.
- **Prover:** When a verifier asks for a proof of possession of a credential, the credential recipient plays the role of the credential prover.
- **Verifier:** This can be any online service whose resources the user wants to access. In order to get this access, the online service will verify the user’s credentials.

The following figure maps the components of the INCOGNITO project to the components of the Idemix protocol:

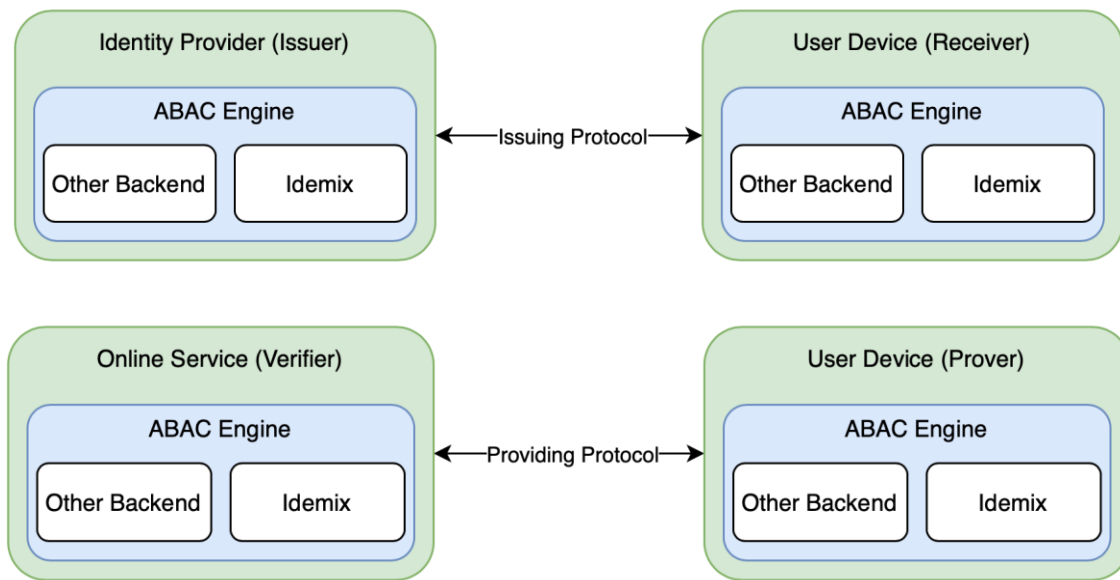


Figure 22. Mapping of INCOGNITO components to the roles of the Idemix Protocol

4.3.1.1 Idemix Protocol

Idemix has two main entities which are playing the four roles mentioned in the section above. There are users who own the credentials and prove their identity to a second entity represented by organizations which issue and verify these credentials. The basic flow of Idemix happens as described next: A user U can obtain a credential C from an issuing organization O_I and show the credential C to a verifier organization O_V . This credential is issued under a pseudonym N at the issuing organization; therefore, the credential is not associated with the user’s real name. Each credential can contain attributes from which the user can choose which of them to share with the verifier organization.

Idemix involves multiple protocols: pseudonym registration, credential issuance and credential verification. It is required 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. As a consequence, a user can choose a master secret S_U which binds together all pseudonyms and credentials. Issuing and verifier organization all have a public / private key pair used to create and validate credentials. The issuing organization uses the private key to generate credentials, while the public key is used by the user or a verifier organization to validate the credential.

Idemix offers one notable feature – namely unlinkability. This is based on the zero-knowledge property of the proof a user gives to a verifier. In order to gain access to some resources, the user shows to a verifier a generated credential together with the zero-knowledge proof. The user aims in this way at proving to the verifier two things: that he possesses the signature generated by the issuing organization and that he knows the master secret associated with his pseudonym. Using this proof, the user does not even send the credentials to the verifier. Therefore, in the end the user can show his credentials to a verifier multiple times without them becoming linkable to each other or to a pseudonym, keeping the user anonymous to the verifier organization.

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.

4.3.1.2 System Setup

The anonymous credential system requires general parameters for credential issuing. These parameters are split into two types: system parameters which represent bit lengths and group parameters which define the groups that are used within the underlying cryptographic scheme. In order to participate in the credential issuance, the two entities – issuers and receivers – must generate parameters too.

An issuer establishes a certain format for the credential generation and in order to receive these credentials, a receiver must agree on the format adopted by the issuer. The issuer uses its key pair so as to generate signatures on lists of attributes. The maximum number of attributes contained in a credential is determined by the length of the public key at the issuer and the number of reserved attributes that are not allowed to be issued (i.e., master secret). Once a user chooses a master secret, he is able to generate as many pseudonyms as he wants which will also be unlinkable between them.

4.3.1.3 Credential Issuance

Credential Issuance protocol involves both the issuer organization and the receiver (the user device). When this protocol runs, both entities agree on a signature that will represent the

cryptographic component of the credential that will be owned by the recipient in the end. The zero-knowledge proof ensures that each party is implementing the protocol correctly at any given time. The issuing protocols works in the following way: The user U firstly contacts the issuing organization O_I and establishes a pseudonym N for the credentials yet to be generated. If the issuer considers N an eligible pseudonym to at a credential with a certain attribute, the organization generates the credential C by signing a statement containing the pseudonym and the respective attribute and send the credential C to the user's device. Now the user can show the credential to a verifier organization in order to gain access to some resources. Therefore, the credential proving protocol starts running.

4.3.1.4 *Credential Proving*

Credential Proving protocol is not as interactive as the Credential Issuance protocol. The user has to create a proof of possession of the credential shown to the verifier. Sometimes, the verifier will ask to see more credentials of the same user if it needs proof of multiple attributes that are stored across different credentials. When the issuer corresponds with the verifier, a credential can be shown only once to a verifier and at the same time a new credential is generated for the user.

The most popular Idemix implementations are using an XML credential format. The credential structure is separated from the credential attributes values so that provers and verifiers could transfer structural metadata without disclosing information about the content of the credential. This format enables the verifiers to understand the content of a received credential. Credential structures are published by the issuers and the user has to accept it if he wants to generate a credential at the respective issuer. The credential contains the actual data corresponding to the chosen format, such as the attributes' values, the master secret and are issued to a certain pseudonym chosen by the user. Besides attributes values, credentials can also have options such as one or multi-show. When showing a credential, the user can choose which of the attributes he wants to prove something about and what to prove about them.

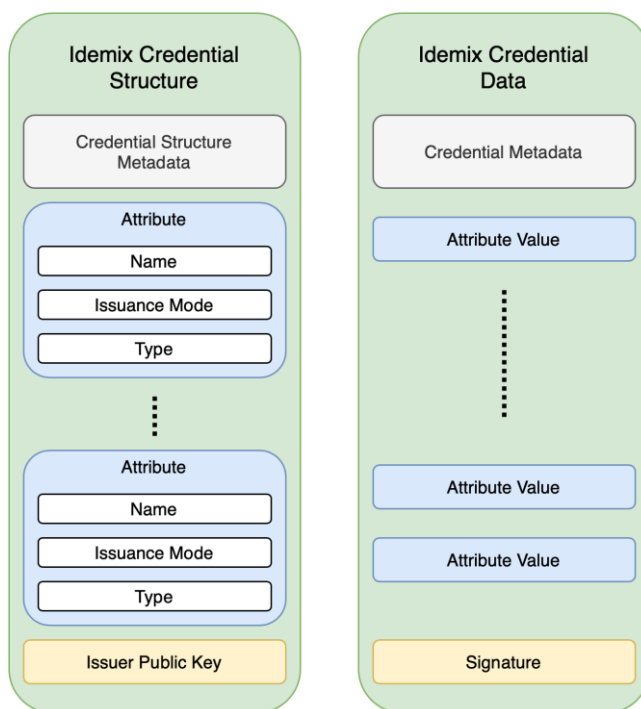


Figure 23. Idemix Credential Structure and Data

4.3.1.5 Attribute Format

Each attribute in a credential structure definition is described by a name, an issuance mode and a type. The name of the attribute is unique within the credential scope and represents a label for the attribute. The issuance mode identifies if the attributes are known to the issuer, if the issuer has a commitment for those attributes and which attributes are hidden from the issuer. The type has four options: string, int, date or enum. The value of an attribute inside the credential is encoded according to its type. When creating a proof of possession of a certain credential, the user can choose which attributes to reveal. Therefore, there are two types of attributes inside a credential: revealed and unrevealed attributes.

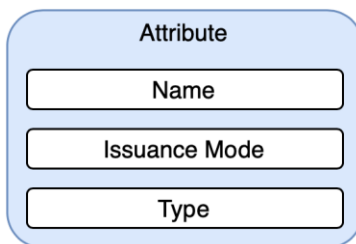


Figure 24. Attribute Structure inside a Credential

4.3.2 Attribute-Based Encryption – OpenABE

Sahai Waters (2004) published a system called “Fuzzy Identity-Based Encryption”, or better known as Attribute-Based Encryption (ABE). This solution allows for multiple private keys to be used with a single public key (hence the “fuzzy” in the title, which can be interpreted as “approximate matching” in computer science). Furthermore, the public keys are constructed from a list of attributes instead of an identity. Anyone who has all the attributes can read the message.

OpenABE is found as a C/C++ software library that contains numerous schemes regarding attribute-based encryption (ABE) along with multiple cryptographic options (e.g., symmetric/asymmetric key encryption, digital signatures, certificates, etc.). OpenABE is a good solution for developers that intend to implement ABE, as it provides:

1. Resistance to collisions, which means that it is near to impossible to find two different inputs that are going to produce the same output when they are hashed.
2. Resistance to “Chosen Ciphertext Attack (CCA)”, which means that malicious entities that perform actions targeting the tampering of data will be in vain.
3. Flexible format regarding the depiction of attributes, as the identity attributes can be represented by any kind of string with no limitations.

OpenABE supports role-based ABE schemes, which is what we are going to utilize in INCOGNITO. OpenABE also supports efficient management for storing and selecting private ABE keys for decryption of ABE ciphertexts.

4.4 Blockchain

The blockchain technology will be integrated at the INCOGNITO platform with the goal of immutably logging:

- Access rights that users give regarding their identity attributes
- IdP/IDC actions regarding users’ identity attributes they share with Service Providers
- Identity attributes saved on the blockchain

INCOGNITO will make use of a permissioned blockchain, adding a second layer of security in order to ensure that the information logged on the distributed ledger, even if it is anonymized, cannot be accessed by the public or entities that do not need to have access to it. All the entries in the distributed ledger are transparent to the involved parties of the transactions and hold them accountable for their actions.

4.4.1 Hyperledger Fabric

The Hyperledger Fabric¹⁸ framework gives developers the ability to build on top of a distributed ledger technology, that is open source and mainly targeted at enterprise solutions. Hyperledger

¹⁸ <https://www.hyperledger.org/use/fabric>

Fabric can be adapted and configured according to the needs and the architecture in which it will be integrated to. If needed, Fabric is versatile and supports a variety of programming languages that can be used to implement smart contracts, i.e. Golang¹⁹, Java²⁰ and Node.js²¹, providing a very big level of adaptability for the developers. Smart contracts (called chaincodes in Fabric) are defined as applications deployed on the blockchain network and are executed by the participating nodes. Moreover, the Fabric platform focuses on permissioned blockchains, which is the technology that we are going to utilize in INCOGNITO. Different roles will be assigned to the blockchain participants with the corresponding access rights. Hyperledger Fabric also has the advantage of being compatible with a number of consensus algorithms among which the developers can choose, making it easier to adopt solutions that do not require a native cryptocurrency to reward miners. This results in threats reduction as malicious actors are not motivated by any valuable asset that they could potentially acquire through illegal actions.

4.4.2 Ethereum

Ethereum²² is a generic blockchain platform developed under the Ethereum Foundation, with its own native cryptocurrency, Ether. It carries common characteristics found in other cryptocurrencies, i.e., digital nature, decentralized and not controlled by a central entity and used to make payments across the globe. Moreover, the Ethereum platform give the ability to developers to create applications that can be deployed on this specific blockchain infrastructure. We are referring to smart contracts that can be utilized in a plethora of use cases. The smart contracts are developed with the use of Solidity, a language designed specific for this purpose. The consensus algorithm used by Ethereum is currently Proof-of-Work (PoW), but there are plans to migrate to a less resource-consuming solution like Proof-of-Stake (PoS). Access to smart contracts is performed at individual's level and can be performed based on either ABAC or RBAC. Lastly, Ethereum and its decentralized applications are all based on a public blockchain, which means that the information logged on the corresponding distributed ledger is available to all, not just the involved/interested parties.

4.4.3 Quorum

Quorum²³ is an open source blockchain platform based on Ethereum, with the difference that it has been optimized to target enterprise solutions. Quorum addresses the privacy issue mentioned in Ethereum, as it is not public and available for everyone to look at the logged transactions. Nonetheless, the access to certain resources remains ABAC or RBAC-based, same as the Ethereum platform. Also, the consensus mechanism options have been broadened, as it supports pluggable consensus algorithms, Raft consensus and pBFT. The smart contracts execution on the blockchain is of significant importance; solidity is used as well for their implementation, which may pose some adjustment difficulties for the developers.

¹⁹ <https://golang.org/>

²⁰ <https://www.java.com/en/>

²¹ <https://nodejs.org/en/>

²² <https://ethereum.org/>

²³ <https://www.goquorum.com/>

4.4.4 R3 Corda

Corda²⁴ is an open source blockchain platform that makes it possible for businesses to perform transactions without any middle-men, preserving privacy. Smart contracts are utilized in order to log information regarding transaction costs and business operations in general. Developers are not confined to a custom programming language for the smart contracts’ implementation, as Kotlin and Java are available to choose from. R3 Corda is also flexible regarding the consensus algorithms that can be incorporated; pluggability is a feature that many developers appreciate, while at the same time Trusted Solo and Raft are provided. Access control is defined in relation to the organization each user belongs to.

Below we can observe a summary of the characteristics of the aforementioned blockchain platforms.

	Hyperledger Fabric	Ethereum	Quorum	R3 Corda
Privacy	Permissioned	Public	Permissioned/ Private	Permissioned
Consensus	<ul style="list-style-type: none"> • Pluggable • Trusted Solo • Kafka 	<ul style="list-style-type: none"> • Proof of Work 	<ul style="list-style-type: none"> • Pluggable • Raft • pBFT 	<ul style="list-style-type: none"> • Pluggable • Trusted Solo • Raft
Access control	Organization level access control (also ABAC & RBAC – based)	ABAC & RBAC – based	ABAC & RBAC – based	Organization level access control
Programming language	<ul style="list-style-type: none"> • Golang • Java • NodeJS 	Solidity	Solidity	<ul style="list-style-type: none"> • Java • Kotlin

Table 2. Summary of the characteristics of the studied blockchain platforms

4.4.5 INCOGNITO blockchain Network

In order to perform the actions described above on the blockchain, the network is going to be comprised by the IDC and the Service Providers. These entities will be able submit transactions on the distributed ledger and perform queries if needed in order to access old entries. Every participant of the blockchain network will maintain an instance of the distributed ledger locally. Below we describe the actions in more detail:

- The user logs in to his/her personal account on the IDC. Through the UMA integration, the user is able to set a policy regarding sharing a set of identity attributes with a group of SPs. That policy and access rights are submitted to the blockchain in the form of transactions. The entity that performs the submission is the IDC, as soon as the user finalizes the changes to the access rights on the profile.

²⁴ <https://www.r3.com/corda-platform/>

- When a user tries to utilize a service from an SP, the SP communicates with the IDC in order to determine whether the identity attributes of the user are adequate to gain access to the provided service. In response, the IDC relays a corresponding message, revealing the user’s identity attributes needed by the SP, in case the user is indeed eligible to utilize the service. The corresponding message is submitted to the distributed ledger by the SP, signed by both the sender (IDC) and the receiver (SP). Note that the identity attributes may not be implicitly revealed, but if a requirement is for example that the user must be over 18 years old, and the IDC informs the SP that this requirement is met, then it is automatically deducted that the user is over 18.
- The IDC will submit to the distributed ledger the values of the users’ identity attributes. We provide an extra layer of security and we ensure that the entities participating on the blockchain network will not be able to access the users’ attributes, as they will be encrypted. The identity attribute values are submitted to the blockchain in order to give the SPs the ability to verify that identity attributes they receive from the IDC have not been tampered with or been subjected to a man-in-the-middle attack. This way the validity of the identity attributes utilized to make use of an online service is ensured.

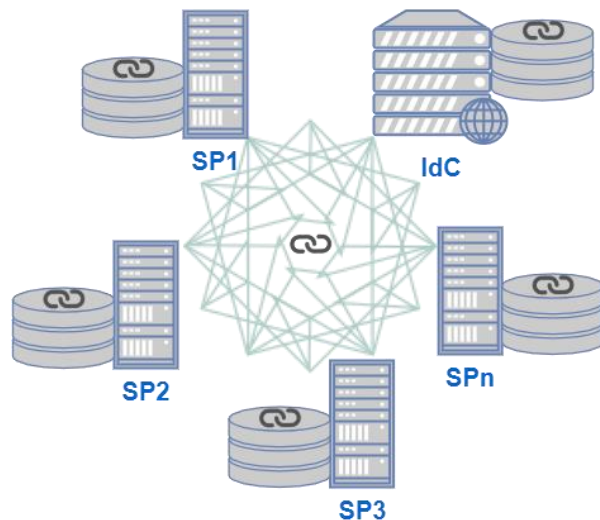


Figure 25. Blockchain network

The consensus algorithm that is going to be used for the blockchain implementation is the practical Byzantine Fault Tolerance (pBFT). The consensus algorithm is utilized in order to successfully submit entries on the distributed ledger by the network participants, which in this case are the IDC and the SPs. More specifically, Byzantine fault tolerance is achieved when the participants of the network that operate correctly reach an agreement on their values. A pBFT network consists of n nodes; there is an upper limit of malicious or malfunctioning number of nodes that can be tolerated by a pBFT network. This number is defined as a constant “ f ”, that is equal to one third of the nodes in the network. If more than $1/3$ of the network nodes are not operating correctly, the pBFT algorithm will not work correctly.

pBFT is energy efficient as it does not make use of complex mathematical computations, compared to Proof-of-Work (PoW) algorithm, which results in reduced use of computational resources and

electricity consumption. An additional advantage over PoW is that the transactions do not require confirmation from numerous different nodes after they have been agreed upon.

4.5 Near-Field Communication (NFC)

Near field communication (NFC), is a form of contactless communication between devices like smartphones or tablets. Contactless communication allows a user to wave the smartphone over an NFC compatible device to send information without needing to touch the devices together or go through multiple steps setting up a connection.

Near field communication maintains interoperability between different wireless communication methods like Bluetooth and other NFC standards. Founded in 2004 by Sony, Nokia, and Philips, the forum enforces strict standards that manufacturers must meet when designing NFC compatible devices. This ensures that NFC is secure and remains easy-to-use with different versions of the technology. Compatibility is the key to the growth of NFC as a popular payment and data communication method. It must be able to communicate with other wireless technologies and be able to interact with different types of NFC transmissions.

In INCOGNITO, the NFC protocol will be part of the Identity Acquisition module. The user will be able to acquire and verify his/her identity attributes from RFID-enabled physical ID documents using his mobile device. These data will be securely stored on the IDC. The developed solution will be in accordance with the GDPR guidelines.

4.5.1 NFC Implementations

1. ReadID

ReadID²⁵ is an NFC ePassport reader application that allows users to scan and verify electronic ID documents and also to read the personal information stored on the document. The application can read modern passports and similar ID documents that have contactless RFID chips installed. ReadID has a number of unique features:

- Establish authenticity of identity documents
- Read personal information from the chip
- Access to high resolution face image on chip
- Optical Character Recognition to scan machine-readable zone (MRZ) and interpret the text for processing
- ReadID SDK's integrated in an any app through APIs

Most of the contemporary smartphones in circulation have NFC functionality. ReadID is available on iOS, from iOS 13, and on Android, from Android 5.0.

²⁵ <https://www.readid.com/about>

2. Mobile ID

Mobile ID²⁶ is a product offered by Gemalto, which is a digital centralized identity service based on Mobile Connect. This solution is based on NFC for acquiring and verifying the users’ identity information from their electronic ID documents.

It offers an identity verification solution that consists of:

- identity document verification
- customer authentication
- risk assessment
- ID verification report

automatic form filling with identity information

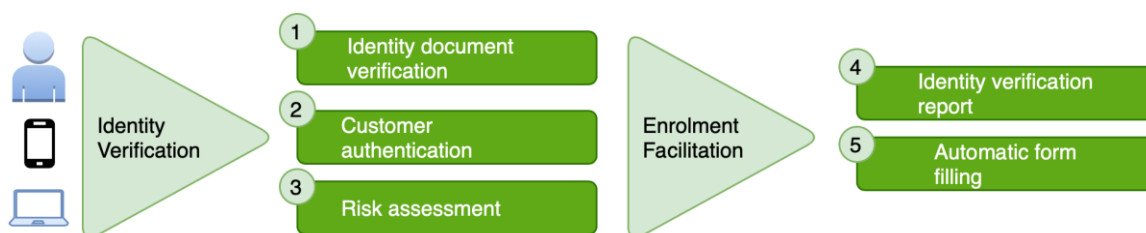


Figure 26. Mobile ID solution and identity verification process

3. Mobile Verify

Mobile Verify²⁷ leverages the NFC protocol in order to verify the authenticity of an ePassport and can be integrated to any application that wants to perform identity verification. During the verification process, they also requesting from the user to capture a selfie, which is compared with the photo on the Identity document.

Mobile Verify’s ID verification engine is a modular cross-platform architecture, which is built on machine learning and advanced computer vision algorithms. In order to achieve the highest accuracy rates, Mitek’s technology was conceptualized to verify the authenticity of an ID document in the following systematic approach:

- Guided document capture
- Document classification
- Data extraction
- Evaluation of authentic elements

²⁶ <http://www.gemalto.com/mobile/id-security/id-verification>

²⁷ <https://www.miteksystems.com/mobile-verify>

4. JMRTD

JMRTD²⁸ is an open-source Java implementation of the Machine-Readable Travel Document (MRTD) standards as specified by the International Civil Aviation Organization (ICAO).

The main features of JMRTD:

- Java API for accessing ICAO compliant eMRTDs and ePassports
- A Java Card eMRTD/ePassport emulator
- Java and Android supported
- Extended access control (EAC) and supplemental access control (SAC/PACE) supported
- LDS 1.7 decoding and encoding
- CBEFF datagroups fully supported
- JPEG2000 and WSQ encoded biometric images supported

	ReadID	Mobile ID	Mobile Verify	JMRTD
Installation /Configuration	One executable application	One executable application	One executable application	Java software/library
Open Source	No	No	No	Yes
Supported Smartphones / Platforms	From iOS 13 and Android 5.0	iOS and Android	iOS and Android	Windows, Linux, Mac OS X, and Android
Development Activity	Active	Active	Active	Active
Documentation	Detailed	Not Available	Not Available	Detailed

Table 3. Comparison between available NFC-based Identity Acquisition implementations

Examining all the details presented in the above table, JMRTD seems to be the best solution to suit the INCOGNITO project's needs.

4.6 WebRTC

With WebRTC, you can add real-time communication capabilities to the application that works on top of an open standard. It supports video, voice, and generic data to be exchanged directly between two or more peers, allowing developers to build powerful voice- and video-communication solutions. The technology is available on all modern browsers as well as on native clients for all major platforms. The technologies behind WebRTC are implemented as an open web standard and available as regular JavaScript APIs in all major browsers. For native clients, like Android and iOS applications, a library is available that provides the same functionality. The WebRTC project is open-source and supported by Apple, Google, Microsoft and Mozilla, amongst others.

²⁸ <http://www.jmrtid.org/about.shtml>

INCOGNITO will utilize the WebRTC protocol, which provides browsers and mobile applications with Real-Time Communications (RTC) capabilities via simple APIs. This solution offers remote identity verification that allows a user to verify the identity of another user online as if they have met in person.

To acquire and communicate streaming data, WebRTC implements the following APIs:

- **MediaStream:** gets access to data streams, such as from the users’ camera and microphone
- **RTCPeerConnection:** audio or video calling, with facilities for encryption and bandwidth management
- **RTCDataChannel:** peer-to-peer communication of generic data

Moreover, there are several ways a real-time communication application or plugin might compromise security. For example, first, an undecrypted media or data might be intercepted en-route between browser, or between a browser and a server. Second, an application might record and distribute video or audio without the user knowing. Third, malware or viruses might be installed alongside an innocuous plugin or app.

Thus, WebRTC has several features to avoid these problems:

- WebRTC implementations use secure protocols such as Datagram Transport Layer Security (DTLS) and Secure Real-time Transport Protocol (SRTP)
- Encryption is mandatory for all WebRTC components, including signaling mechanisms
- WebRTC is not a plugin, its components run in the browser sandbox and not in a separate process, components do not require different installation, and are updated whenever the browser is updated.
- Camera and microphone access must be granted explicitly and, when the camera or microphone is running, this clearly shown by the user interface.

4.7 Tor Network

Tor is free and open-source software for enabling anonymous communication. Tor directs Internet traffic through a free, worldwide, volunteer overlay network consisting of more than seven thousand relays to conceal a user's location and usage from anyone conducting network surveillance or traffic analysis. Using Tor makes it more difficult to trace Internet activity to the user: this includes "visits to Web sites, online posts, instant messages, and other communication forms". Tor's intended use is to protect the personal privacy of its users, as well as their freedom and ability to conduct confidential communication by keeping their Internet activities unmonitored.

In INCOGNITO, the Tor Network will be implemented in the User’s Device (using a Tor Network library), offering a second layer of security on top of the Idemix protocol. Tor helps to reduce the risks of both simple and sophisticated traffic analysis by distributing the user's transactions over several places on the Internet so that no single point can link his destination. Instead of taking a direct route from source to destination, data packets on the Tor network take a random pathway through several relays that cover the user's tracks so no observer at any single point can tell where the data came from or where it's going.

To create a private network pathway with Tor, the Tor software incrementally builds a circuit of encrypted connections through relays on the network. The circuit is extended one hop at a time, and each relay along the way knows only which relay gave it data and which relay it is giving data to. No individual relay ever knows the complete path that a data packet has taken.

Once a circuit has been established, many kinds of data can be exchanged, and several different sorts of software applications can be deployed over the Tor network. Because each relay sees no more than one hop in the circuit, neither an eavesdropper nor a compromised relay can use traffic analysis to link the connection's source and destination.

5 Conclusions

The purpose of this deliverable was to define the reference architecture of the INCOGNITO project. In this document, we first defined the reference architecture of the INCOGNITO platform and we provided a detailed description of all the architectural components which are: 1) User Device; 2) Identity Consolidator; 3) Decentralized Identity Management Blockchain; 4) Identity Provider; and 5) Service Providers. In addition, we also provided a description of all the building blocks of the INCOGNITO framework, while we also provided details about the main architectural components that are involved to form each building block of INCOGNITO. In the end, we also provided a description of all the protocols and technologies that will be used for the implementation of this architecture in the individual work packages, and how each protocol is used and enhanced in INCOGNITO.

We note that, the purpose of this deliverable was to define the architectural design of the INCOGNITO platform and provide a general description of the how the components of the architecture will communicate with each other, as well as a description of the protocols that will be used and enhanced in order to facilitate the implementation of each architectural component and the interactions between them. As a result, the outcome of this work is the starting point and will guide the design and implementation of the individual architectural components in the individual work packages of the project (WP3, WP5, WP6). More precisely, this document will guide the implementation of WP3 and its respective architectural components and modules that together will form a Qualified Anonymity framework. In addition, this deliverable provided all the required details that are needed for the implementation of an Identity Acquisition, integration and management platform, a user-friendly Consent Management platform, and the Identity Consolidator component in general in the context of WP4. Last, in this deliverable we also included a description of the components and modules that together will form the Advanced UI/UX AI-based assistant that will be implemented in the context of WP5. A more detailed description about the functionalities, as well as all the implementation details of each individual architectural component will be provided in the corresponding future deliverables of WP3, WP4, and WP5.

Last, we strongly believe that the reference architecture of INCOGNITO is able to address all the weaknesses of today's Web authentication schemes and deem the traditional password obsolete, while also offering a privacy-preserving solution for device-centric and attribute-based

authentication. Through the work performed in the context of this deliverable we were also able to incorporate a qualified anonymity framework within such a solution that allows users to preserve their privacy and remain untraceable while accessing services on the Web. At the same time the defined Identity acquisition, integration and management platform allows users to consolidate their multiple soft proofs of their fragmented online and real-world identities fast into independent verifiable identity attributes that are stored at a decentralized identity management blockchain network. In addition, the Advanced AI-based assistant and the user-friendly Consent Management platform defined in this document enables users manage their identity and their privacy in the level and way they desire.

Finally, we note that the architecture of the INCOGNITO platform and all its key aspects have been well disseminated to the scientific community through the submission of a [scientific journal](#) which has been accepted for publication at the IEEE Transactions on Information Forensics and Security (TIFS)²⁹. This scientific publication is also available in the list of publications of the INCOGNITO website³⁰.

²⁹ <https://incognito.socialcomputing.eu/wp-content/uploads/2019/publications/08931622.pdf>

³⁰ <https://incognito.socialcomputing.eu/publications/>

6 References

- [1] M. P. Machulak, E. L. Maler, D. Catalano, and A. Van Moorsel, “User-managed access to web resources,” 2010, doi: 10.1145/1866855.1866865.
- [2] J. Camenisch and A. Lysyanskaya, “An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation,” in *Advances in Cryptology --- EUROCRYPT 2001*, 2001, pp. 93–118.
- [3] M. Sabt, M. Achemlal, and A. Bouabdallah, “Trusted Execution Environment: What It is, and What It is Not,” in *2015 IEEE Trustcom/BigDataSE/ISPA*, Aug. 2015, vol. 1, pp. 57–64, doi: 10.1109/Trustcom.2015.357.
- [4] R. Want, “Near field communication,” *IEEE Pervasive Comput.*, vol. 10, no. 3, pp. 4–7, Jul. 2011, doi: 10.1109/MPRV.2011.55.
- [5] E. Yuan and J. Tong, “Attributed based access control (ABAC) for Web services,” in *IEEE International Conference on Web Services (ICWS’05)*, Jul. 2005, p. 569, doi: 10.1109/ICWS.2005.25.
- [6] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” 2019.
- [8] J. Camenisch and E. Van Herreweghen, “Design and implementation of the idemix anonymous credential system,” 2002, doi: 10.1145/586110.586114.