Marie Sklodowska Curie, Research and Innovation Staff Exchange (RISE)

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

WP4 – Identity acquisition, integration and management
D4.1 “Identity Acquisition and Integration Platform”

Editor(s):	CUT
Author(s):	Kostantinos Papadamou (CUT - outside secondment), Markos Charalambous (CUT - outside secondment), Petros Papagiannis (CUT - secondee), Ioana Stroinea (CSGN - secondee), Nikolaos Salamanos (CUT - secondee), Vaios Bolgouras (UPRC - outside secondment), Evangelos Kotsifakos (LST - outside secondment), George Kalatzantonakis (LST - secondee), Nikolaos Episkopos (FOGUS - secondee), Nicolas Kourtellis (TID - secondee), Nikos Passas (UPRC - outside secondment), Christos Xenakis (UPRC - secondee), Angeliki Panou (UPRC - secondee), Michael Sirivianos (CUT - outside secondment)
Dissemination Level:	Public
Nature:	Report
Version:	3.0

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 / Commision is not responsible for any use that may be made of the information it contains.

Project Profile

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

Partners

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

Document History

Version	Date	Author(s)	Remarks
0.1	30/08/2020	Kostantinos Papadamou (CUT), Petros Papagiannis (CUT)	Executive Summary and Table of Contents
0.2	04/09/2020	Kostantinos Papadamou (CUT), Vaios Bolgouras (UPRC), Angeliki Panou (UPRC)	Introduction
0.3	20/09/2020	Kostantinos Papadamou (CUT), Vaios Bolgouras (UPRC), Ioana Stroinea (CSGN)	Identity Acquisition and Integration Platform
0.4	22/09/2020	Kostantinos Papadamou (CUT), Markos Charalambous (CUT), Petros Papagiannis (CUT)	Physical Identity Acquisition Module
0.5	15/10/2020	Petros Papagiannis (CUT), Nicolas Kourtellis (TID), Nikolaos Episkopos (FOGUS)	Online Identity Acquisition Module
0.6	18/10/2020	Vaios Bolgouras (UPRC), Ioana Stroinea (CSGN), Petros Papagiannis (CUT)	Identity Integration and Normalization Module
0.7	25/10/2020	Kostantinos Papadamou (CUT), Markos Charalambous (CUT), Vaios Bolgouras (UPRC), Ioana Stroinea (CSGN)	Identity Attributes Storage schema definition and description
0.8	10/11/2020	Ioana Stroinea (CSGN), George Kalantzantonakis (LST), Nikolaos Salamanos (CUT)	Identity Attributes Storage REST API definition and documentation
0.85	25/11/2020	Ioana Stroinea (CSGN), George Kalantzantonakis (LST)	IDC 3rd Party REST API definition and documentation
0.9	10/12/2020	Kostantinos Papadamou (CUT), Nikolaos Salamanos (CUT), Nicolas Kourtellis (TID), Nikolaos Episkopos (FOGUS)	Update of the Identity Attributes Storage REST API and of the IDC 3rd Party REST API documentations and writing of Conclusion
0.95	20/12/2020	Kostantinos Papadamou (CUT), Michael Sirivianos (CUT), Christos Xenakis (UPRC), Evangelos Kotsifakos (LST)	Document Review
1.0	27/12/2020	Nikos Passas (UPRC)	Document Submission
2.0	22/01/2021	Nikos Passas (UPRC)	Edited based on the comments from the EU
3.0	15/02/2021	Nikos Passas (UPRC)	Edited based on the comments from the EU

Deliverable Dates

Delivery	Date
Contract delivery due date	31/12/2020 (M24)
Actual delivery date of version n.1	31/12/2020 (M24)
Actual delivery date of version n.2	22/01/2021 (M25)
Actual delivery date of version n.3	15/02/2021 (M26)

Fellow ID	Researcher Name	Researcher Category	Declaration No.	Person Months
8	Christos Xenakis	ER	7	0.3
9	Angeliki Panou	ESR	6	0.3
11	Ioana Stroinea	ESR	9, 18	2.63
5	Petros Papagiannis	ESR	4	4
14	George Kalantzantonakis	ESR	14	4
18	Nikolaos Salamanos	ER	15	3
21	Nicolas Kourtellis	ER	17	2
22	Nikolaos Episkopos	TS	19	3

Executive Summary

This current deliverable describes the design and implementation of the Identity Acquisition and Integration platform, one of the key components of the INCOGNITO platform and is part of the activities of Work Package 4. 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 the INCOGNITO body 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 (e.g., an AI-based assistant) without the need to disclose their full identity.

D4.1 tackles the objectives of Task 4.1: "Identity Acquisition of multiple soft proofs of identities and Identity integration engine" via the scientific research activities reported in this deliverable. More specifically, the purpose of this deliverable is to specify the techniques used and the solutions that have been designed and implemented in order to provide an Identity Acquisition and Integration platform that enables the acquisition and the seamless integration of the multiple soft proofs of identities of a user, both the real-world identities (e.g., national ID cards, utility bills, ePassports, eID cards, etc.), as well as all the online accounts of the user, through web and mobile application interfaces.

Furthermore, the Identity Acquisition and Integration is part of the broader Identity Consolidator, which is a complete identity acquisition and management and a consent management solution that enables users to manage their identity and their consent, while at the same time it provides access to this information to third parties (e.g., Service Providers) taking into account all the relevant security, privacy, authorization and authentication aspects.

Finally, this document presents the entire design and implementation of the Identity Acquisition and Integration platform and it carefully details all the supported functionalities offered by all the components that together form this platform and offer to the user a user-friendly and secure platform for identity acquisition, validation and integration. At the same time, this deliverable also defines and documents the implementation of other critical components and modules (e.g., RESTful APIs) that are required for the proper integration between the diverse modules of the Identity Consolidator and other components of the INCOGNITO platform.

Please note that hereinafter, mentions of "INCOGNITO" refer to the platform that will be implemented in the context of the project, unless stated otherwise (for example "INCOGNITO body/consortium/users etc.").

Table of Contents

Document History ***3***

Deliverable Dates ***4***

Executive Summary ***5***

Table of Figures ***9***

Table of Abbreviations ***10***

1 Introduction ***11***

1.1 General Overview of the Identity Acquisition and Integration Platform **11**

2 Identity Acquisition and Integration Platform ***12***

2.1 Identity Consolidator Platform & Identity Acquisition and Integration Web/Mobile Application **14**

3 Physical Identity Acquisition and Verification Module ***16***

3.1 Physical Identity Acquisition **17**

3.1.1 Initiate Physical Identity Acquisition process 18

3.1.2 Capture Physical Identity Document and User Face Pictures 19

3.1.3 View and Manage Physical Identity Document Information and Uploaded Pictures 20

3.1.4 User Addresses' Information Acquisition 20

3.1.5 RFID-enabled Identity Document Scanning using NFC-enabled devices 21

3.2 Physical Identity Verification **26**

3.2.1 Peer-to-Peer Verification Using Crowdsourcing Techniques 26

3.2.2 Automated Verification Techniques 28

3.2.3 Remote Identity Verification using the WebRTC Protocol 29

3.3 Physical Identity Acquisition Mobile Application **30**

3.4 Physical Identity Acquisition Module Internal REST API **31**

3.4.1 Physical Identity Document Verification Status 32

3.4.2 Identity Attribute Verification Status 33

3.4.3 Store RFID-enabled Physical Identity Document Information 33

4 Online Identity Acquisition Module ***35***

4.1 User-assisted Online Identity Acquisition process **36**

4.2 Automatic Online Identity Acquisition process **37**

5 Identity Integration and Normalization ***38***

5.1 Identity Integration and Normalization Algorithm **38**

6 Identity Attributes Storage ***39***

6.1 Notation and Conventions **40**

6.2 User Account Table **40**

6.3 User Physical Identities Info Table **41**

6.4 Identity Providers Data Table 42
6.5 User Names Info Table 44
6.6 User Phones Info Table 46
6.7 User Addresses Info Table 47
6.8 MediaItem Table 49
6.9 Verification Table 50
6.10 Audits Info Table 51
6.11 Cryptographic Credentials Table 52
6.12 Service Providers Users Table 53
6.13 Identity Providers Users Table 53
6.14 Identity Providers Metadata Table 54
6.15 User Physical Identity Documents Cropped Regions Table 54
6.16 User Relationships Info Table 55
7 Identity Attributes Storage REST API 55
7.1 REST Services 56
7.2 HTTP Request & Response Header Fields 56
7.3 Request Operations 57
7.3.1 Creation of Information 57
7.3.2 Conditional Requests 57
7.3.3 Full Vs Partial Modifications 57
7.3.4 Delete Information 58
7.4 Query Invocations 58
7.4.1 Authentication & Authorization 58
7.4.2 Query Parameters 59
7.4.3 Data Filtering 59
7.5 Response Format 61
7.6 Error Codes 61
7.7 Structure 61
7.7.1 Singular Fields 61
7.7.2 Plural Fields 62
7.7.3 Name Element 63
7.7.4 address Element 63
7.7.5 account Element 63
7.7.6 verificationInfo Element 64
7.7.7 Other Elements 64
8 Identity Consolidator (IDC) 3rd Party REST API 65
8.1 High-Level Operations 65

8.2 Transfer Trust Between Service Providers 65
8.3 Delete User Identity Consolidator Account 66
9 Conclusion 67
10 References 69

Table of Figures

Figure 1. Identity Acquisition and Integration Platform 13
Figure 2. Identity Consolidator platform Web application main page 15
Figure 3. Identity Consolidator Mobile Application 16
Figure 4. Declare Physical identity document information 19
Figure 5. List of required parts to crop from the captured identity document photo 20
Figure 6. Address information acquisition by declaring its details 21
Figure 7. RFID-enabled Identity document verification guidelines (left) and prompt to choose the document to be verified (right) 24
Figure 8. Enter Identity document required information for BAC authentication 24
Figure 9. Data from the biometric passport has been read. Data has been anonymized for privacy reasons 25
Figure 10. Peer-to-peer audit of the identity document number of a user 28
Figure 11. Screen where the verifier declares the email address of the user that wants to be verified using remote identity verification 30
Figure 12. Remote Identity Verification using the WebRTC protocol. Faces hidden for privacy reasons 30
Figure 13. Screenshots of from some pages of the Physical Identity Acquisition Mobile Application 31
Figure 14. Main page of the Online Identity Acquisition Web application 36
Figure 15. Main page of the Online Identity Acquisition Mobile application 37

Table of Abbreviations

AA	Active Authentication
API	Application Programming Interface
BAC	Basic Access Control
HTTP	Hypertext Transfer Protocol
ID	Identity
IDC	Identity Consolidator
IdP	Identity Provider
IIM	Identity Integration Module
IoT	Internet of Things
JSON	JavaScript Object Notation
NFC	Near Field Communication
OAuth	Open Authentication
OCR	Optical Character Recognition
OIDC	OpenID Connect
REST	Representation State Transfer
RFID	Radio Frequency Identification
SSL	Secure Sockets Layer
SP	Service Provider
TLS	Transport Layer Security
UD	User Device
UI	User Interface
UX	User Experience
XML	eXtensible Markup Language

1 Introduction

The Identity Acquisition and Integration platform is among the major components of the INCOGNITO architecture. This platform plays a major role in most of the piloting activities and use cases of the INCOGNITO platform since it facilitates the seamless integration of the multiple soft proofs of identities (both physical and online) of a user. The purpose of this document is to provide all the details about the architectural design and implementation of the Identity Acquisition and Integration platform and the broader Identity Consolidator (IDC) platform of INCOGNITO. The IDC platform is central to INCOGNITO's architecture and will integrate all the modules that will be implemented in the context of WP3, WP4, and WP5.

First, we provide a generic description of the architectural design of the Identity Acquisition and Integration platform and the other modules of the IDC that together form this platform and enable the acquisition and integration of the multiple identity attributes of a user. Next, following the defined architecture, we provide a detailed description of the implementation and all the functionalities supported by the platform. Last, we also provide a description of the Identity Attributes Storage schema, which holds the acquired and normalized identity information of the users.

We note that the implementation of some of the modules of the Identity Acquisition & Integration platform and the Identity Consolidator in general, reported in this deliverable, was initiated during the ReCRED project and have also been reported in Deliverable 12 of ReCRED. However, we clarify that in the context of Task 4.1 of INCOGNITO we have worked across improving the security, User Experience (UX), performance, deployability, and scalability of these modules. At the same time, we have implemented all the new functionalities required by INCOGNITO such as the implementation of: 1) active authentication of the RFID-enabled identity document scanning using NFC, which allow us to identity and defend against cloned identity documents; 2) a more reliable identity integration and normalization process; 3) acquisition of biometric features from biometric identity documents; and 4) a remote Identity verification process, leveraging the WebRTC protocol, which increases the confidence score and credibility of a user on the platform. Moreover, in the context of INCOGNITO and Task 4.1, we have also performed all the required modifications on the back end of the Identity Consolidator that will facilitate the implementation of the modules and functionalities defined by tasks T4.2 and T4.3 of WP4, as well as their integration with the Identity Consolidator platform. More details about the specific work done for each module can be found in the beginning of each section.

Last, it also worth noting that, the implementation of the modules of the IDC is a work in progress as the effort defined in T4.2 and T4.3 requires the continuous development and enhancement of the backend and front-end of the modules reported in this deliverable. For instance, we will continue updating the schema of the Identity Attributes Storage, the Identity Attributes Storage REST API, and the IDC 3rd Party API for the purposes of T4.2 and T4.3.

1.1 General Overview of the Identity Acquisition and Integration Platform

The Identity Acquisition & Integration platform is one of the major components of the INCOGNITO architecture since it facilitates the seamless integration of the multiple soft proofs of identities of a user. More specifically, this platform is responsible for horizontally binding the online identities of a user and vertically binding the physical identities of a user to independent verifiable identity attributes. The Identity Acquisition & Integration platform offers the required web and mobile interfaces for quickly acquiring the multiple diverse identity attributes of a user from his physical identity documents (i.e., national ID card) and online identities (i.e., Facebook). Using the developed interfaces users can voluntarily submit identity attributes and proofs of online account ownership in order to achieve the consolidation of their physical and online identities and have access to additional functionalities related to their submitted identity attributes. The Identity Acquisition and Integration platform consists of the following modules: 1) Physical Identity Acquisition and Verification module; 2) Online Identity Acquisition module; 3) Identity Integration and Attributes Normalization module; 4) Identity Attributes Storage; 5) Identity Attributes Storage REST API; and the 6) IDC 3rd Party REST API.

First, the Physical Identity Acquisition module is responsible for the acquisition and validation of the identity attributes of the user from the multiple soft proof of physical identities (e.g., national ID card, ePassport, utility bills, university certificates, etc.) in a secure and privacy preserving manner. Next, the Online Identity Acquisition module offers the required functionality for the acquisition of the identity attributes of the user from his/her online accounts (such as Facebook, or any other Identity Provider (IdP)). The Online Identity Acquisition module communicates with IdPs using federated authentication protocols (such as OpenID Connect [1] and OAuth [2]). Furthermore, the Identity Integration and Attributes Normalization module is responsible for the normalization and integration of all the identical attributes of the user assigning them a normalized value and a confidence score.

Next, all the acquired, verified, and normalized identity information of the users will be securely stored in the Identity Attributes Storage and all the modules of the Identity Acquisition and Integration platform and the IDC in general will be able to interact with the Identity Attributes Storage through the Identity Attributes Storage REST API. Last, all the other entities of the INCOGNITO architecture like Service Providers and Identity Providers are able to communicate with the IDC for requesting or storing identity attributes of the users using the IDC 3rd Party REST API.

The rest of this document is structured as follows. Section 2 defines the architecture of the first version of the Identity Consolidator, which includes all the modules that are introduced and consist the Identity Acquisition and Integration platform. Section 3 describes the Physical Identity Acquisition and Verification module, while Section 4 provides all the details of the Online Identity Acquisition module. Next, Section 5 describes the Identity Integration and Normalization module. Section 6 provides a description of the schema of the Identity Attributes Storage. Last, Sections 7 and 8 provide the implementation details and the documentation of the Identity Attributes Storage REST API and the IDC 3rd Party REST API, respectively. We conclude in Section 9.

2 Identity Acquisition and Integration Platform

The Identity Acquisition and Integration platform is one of the major components of the INCOGNITO architecture since it facilitates the seamless acquisition, verification, and integration of the multiple soft proofs of identities of a user and online accounts. This platform is responsible for vertically binding all the physical identities of the user and for horizontally binding the online identities of the user into independent verifiable identity attributes. Through the developed Identity Acquisition and Integration web and mobile platforms, users are able to submit proofs of their real-world identities and proofs account ownership and have verifiable identity attributes in the INCOGNITO platform that can later use to access third party services on the Web. Figure 1 below depicts the architecture of the first of the initial version of the IDC which includes all the modules that are part of the Identity Acquisition and Integration platform.

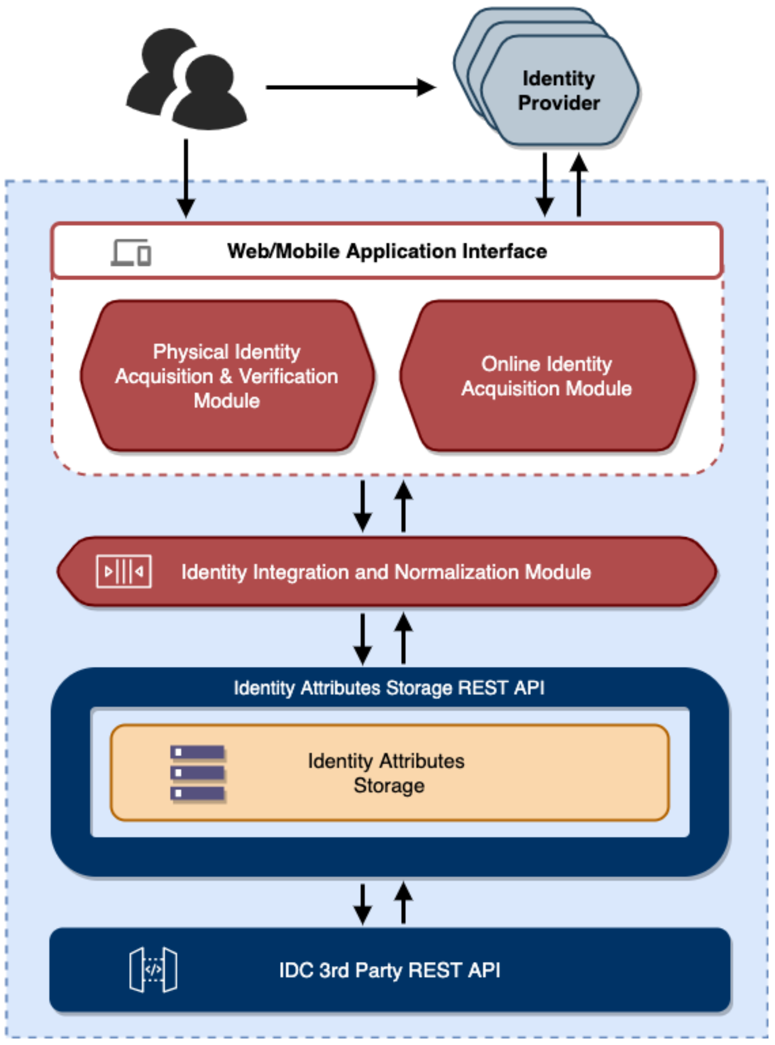

Identity Provider
Web/Mobile Application Interface
Physical Identity Acquisition & Verification Module
Online Identity Acquisition Module
Identity Integration and Normalization Module
Identity Attributes Storage REST API
Identity Attributes Storage
IDC 3rd Party REST API

Figure 1. Identity Acquisition and Integration Platform

Initially, this platform contains the *Physical Identity Acquisition and Verification module*, which is responsible for securely acquiring and verifying all the identity information of a user from his/her real-world identity documents. The functionality of this module is exposed to the users through web and mobile application interfaces, which allow us to leverage smart trusted-computing enabled devices (e.g., mobile phones) to securely acquire all the physical characteristics and the information included in the real-world identities of the users. In addition, this module uses trusted software paths of commodity devices to securely capture, through the device's camera device, pictures of the user and his/her physical identity documents. Identity information of the users (i.e., location) that are used for verification of his identity attributes are captured though the sensors available on commodity mobile devices. Upon acquisition of all the users' identity information, we use various verification algorithms in order to validate all the collected identity information. The identity verification process uses existing techniques (such as OCR) and peer-to-peer (crowdsourcing) techniques. Automated verification is established on the acquired photos of the users using face detection, face recognition and OCR. Peer-to-peer verification using crowdsourcing techniques takes place to verify that the information on the acquired photos match the declared identity information and physical characteristics of the users. All the verified identity information is then stored in the *Identity Attributes Storage* as independent verifiable identity attributes. The Identity Acquisition and Verification module supports additional verification of the collected identity information through remote identity verification by trained professional auditors leveraging the WebRTC protocol (see Section 3.2.3).

We note that, having enough information about a user, the Identity Acquisition and Verification module is in place to infer other identity attributes by aggregating the collected identity information of a user. For example, knowing that a user is between the age of 18 to 22 and he has not yet submitted a bachelor’s degree, we can assume with high confidence that he is still a student at a university.

The horizontal binding of the various online identities of a user (e.g., Facebook, Google+, etc.) is performed by the *Online Identity Acquisition module*. For each online account an online authentication and authorization process using federation authentication protocols (i.e., OpenID Connect) is required so that the user give explicit authorization to the Online Identity Acquisition module to access and retrieve his account's personal information. Following the authentication process, the attributes acquisition takes place, and the collected attributes are stored in the Identity Attributes Storage.

At the same time, the *Identity Integration and Normalization module*, which runs as a background service, is responsible for the normalization of all the collected identity information of the user into independent verifiable distinct identity attributes and for assigning a confidence to each normalized attribute. In other words, it is responsible for aggregating and connecting the acquired online and physical identity attributes of the user, as well as for inferring the veracity of the claimed identity attributes via means of statistical data analysis techniques. The Identity Integration and Normalization module is also responsible for assigning confidence scores for the veracity of the attributes for labelling identity attributes based on their origin. In the end, all the normalized attributes are stored in the Identity Attributes Storage of the IDC.

All the aforementioned modules interact with the Identity Attribute Storage using the *Identity Attributes Storage REST API*. This API enables all the previously described modules to interact with the Identity Attributes Storage for READ, WRITE, and UPDATE the user's identity information. On the other hand, external entities outside the IDC and third parties, are able to communicate with the IDC and its Identity Attributes Storage.

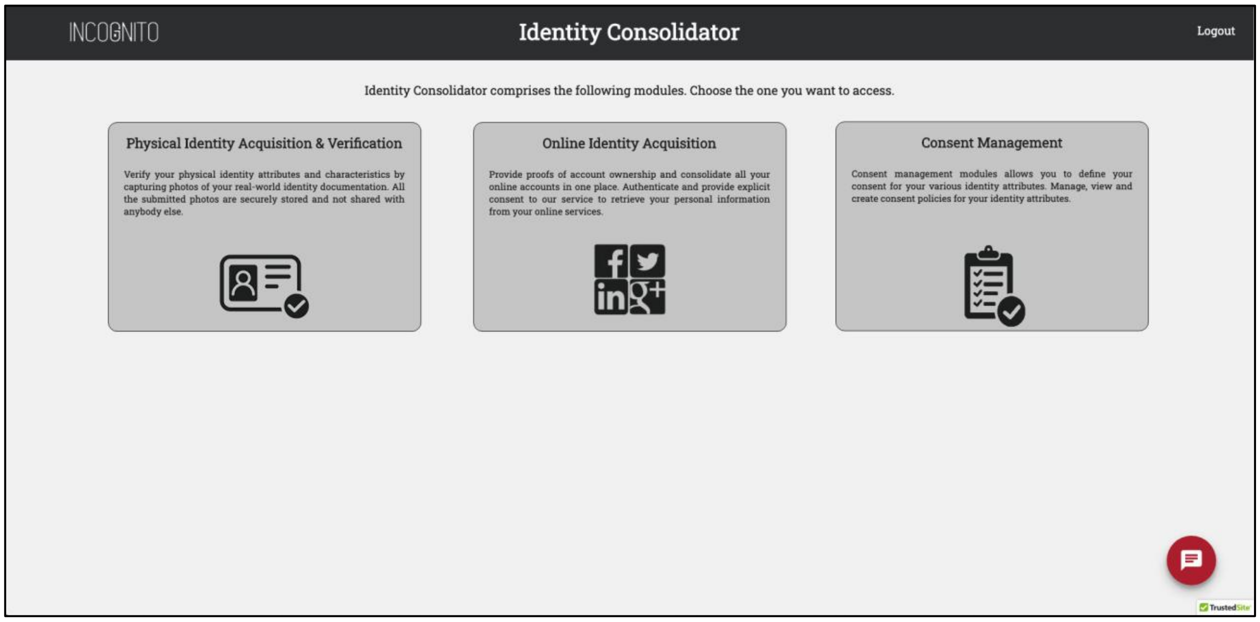
Last, users will be able to authenticate and interact with the IDC and the web and mobile applications of the Identity Acquisition and Integration platform various diverse authentication mechanisms like a FIDO-enhanced OpenID Connect mechanism, which is described in other deliverables of INCOGNITO.

2.1 Identity Consolidator Platform & Identity Acquisition and Integration Web/Mobile Application

The Identity Consolidator platform integrates multiple modules/platforms that together offer to the user a unique identity acquisition, verification, and management solution. These modules are the following: 1) Physical Identity Acquisition and Verification module; 2) Online Identity Acquisition module; 3) Identity Integration and Normalization Module; 4) Consent Management module; and 5) Credentials Management module. Modules (1), (2), and (3) are the core components of the Identity Acquisition and Integration platform. A detailed description about the design and implementation of the other modules will be provided in future deliverables. All the above modules are communicating with the Identity Attributes Storage through the Identity Attributes Storage REST API.

Extending ReCRED. We note that the initial design of the Identity Consolidator platform is influenced by the ReCRED project and its IDC component. Also, the implementation of some of the modules of the Identity Acquisition & Integration platform and the Identity Consolidator initiated during the ReCRED project and have also been reported in Deliverable 12 of ReCRED. However, in task T4.1 of INCOGNITO we built upon ReCRED's IDC design and implementation by improving the security, UX, performance, as well as performance, deployability and scalability using virtualization (i.e., by deploying modules in separate Docker Containers). In addition, we have also completely redesigned and re-implemented some of the modules following the latest software design guidelines and using latest programming language frameworks (e.g., PHP Laravel[1] v.8, Node.js Express.js framework[2]).

In this context, we have developed and deployed a Web and a Mobile application that integrates under one platform all the primary modules of the IDC. This web application has been developed using HTML5, JavaScript ES6, and PHP [3] and is running on an Apache Server [4] at the Cyprus University of Technology datacenter. The communication between the clients and our servers is encrypted and secured as a result of an SSL/TLS certificate that we have installed at our server. Figure 2 depicts the main page of the IDC from where the user is able to access all the modules that the IDC offers, including but not limited to the modules of the Identity Acquisition and Verification platform. Figure 3 shows the respective page in the INCOGNITO mobile application. The user is able to access the main page of the Web and mobile application only if he/she has already authenticated against the Identity Consolidator and a secure session exists.



Identity Consolidator comprises the following modules. Choose the one you want to access.

Physical Identity Acquisition & Verification

Verify your physical identity attributes and characteristics by capturing photos of your real-world identity documentation. All the submitted photos are securely stored and not shared with anybody else.

Online Identity Acquisition

Provide proofs of account ownership and consolidate all your online accounts in one place. Authenticate and provide explicit consent to our service to retrieve your personal information from your online services.

Consent Management

Consent management modules allows you to define your consent for your various identity attributes. Manage, view and create consent policies for your identity attributes.

Figure 2. Identity Consolidator platform Web application main page

[1] https://laravel.com/
[2] https://expressjs.com/

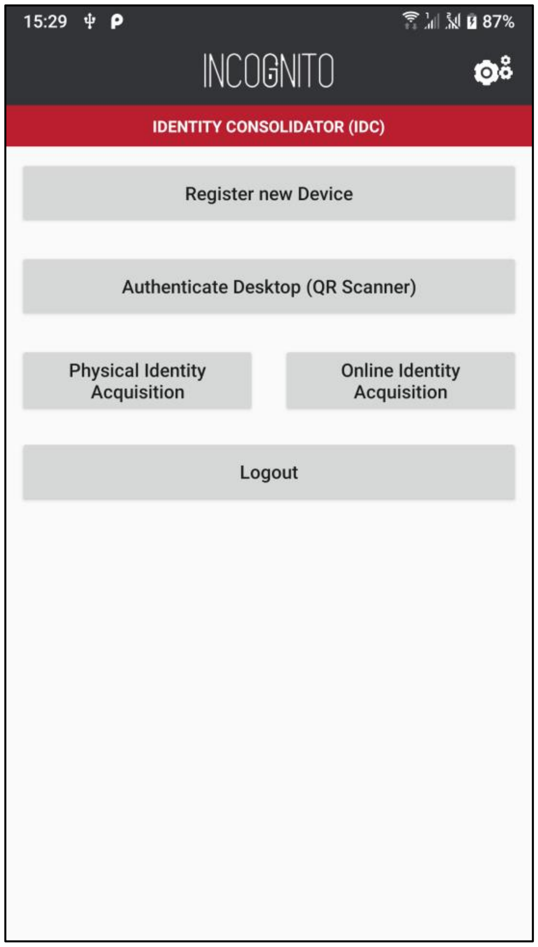

15:29
87%
INCOGNITO
IDENTITY CONSOLIDATOR (IDC)
Register new Device
Authenticate Desktop (QR Scanner)
Physical Identity Acquisition
Online Identity Acquisition
Logout

Figure 3. Identity Consolidator Mobile Application

3 Physical Identity Acquisition and Verification Module

The Physical Identity Acquisition and Verification module of the IDC is responsible for vertically binding the real-world identities of the user. Its main purpose is to collect all the identity information from all the diverse types of the real-world identities of a user (e.g., national ID Card, biometric passport, utility bills, university degree certificate, driving license, etc.), verify them and convert them into independent verifiable identity attributes.

Extending ReCRED. Here it is important to note that the implementation of the Physical Identity Acquisition and Verification module was initiated in the ReCRED project and has been reported in Deliverable 12 of this project. However, we also note that although we kept the same look and feel, in the context of task T4.1 of INCOGNITO, we have re-implemented this module using the latest version of PHP Laravel framework (v.8) [5] with extensive refactoring of the codebase. Furthermore, we have also performed several other security and performance improvements. More specifically, we have made the following notable additions/improvements to the Physical Identity Acquisition and Verification module:

1. We have re-implemented the automated identity verification processes using the latest and more accurate machine learning algorithms for face detection and recognition, such as the one available here, which has an accuracy of 99.38%.
2. In the context of the profession certificate acquisition and verification we have added support for the university certificate of a user which can be used as a proof for his/her qualifications.
3. We have increased the security of the RFID-enabled biometric identity document scanning by implementing Active Authentication [6], which allows to detect and defend against forged or cloned biometric identity documents.
4. Leveraging recent advancements in virtualization we have also deployed the Physical Identity Acquisition and verification module as a separate docker container so that it can easily scale when this is required.

The Physical Identity Acquisition module has been implemented as a web application for desktop computers and as a mobile application. In the rest of this section, we describe in detail all the functionalities supported by the current version of the Physical Identity Acquisition and Verification module as it is central for the understanding of this deliverable.

The Physical Identity Acquisition and Verification module is subdivided into two main processes, the identity acquisition and the identity verification process. The identity acquisition process is the one that requires users’ involvement at most and involves the acquisition of the physical characteristics of the users, as well as the identity information included in their physical identity documentation. This process makes use of the trusted paths on the devices in order to enable the users to securely capture photos of their face and their physical identity documentation. Additionally, the physical characteristics of the users are captured (such as location) by exploiting existing techniques and the available sensors on the devices. All the captured pictures are encrypted and securely stored in a protected directory in the IDC server while the acquired identity information is securely stored in the Identity Attributes Storage through the Identity Attributes Storage API. In general, the physical identity acquisition offers the following: 1) acquisition of all the information included in the real-world identity documents of the user; 2) user physical address information acquisition; 3) management of the collected user's identity information; and 4) acquisition of information included in biometric (RFID-enabled) identity documents using NFC.

On the other hand, the identity verification process uses various secure methods for the verification of all the acquired user's identity information. More precisely, for the identity verification process we employ crowdsourcing techniques and other automated means such as Optical Character Recognition (OCR), face detection and recognition to verify all the collected identity information, identity user pictures, and the physical characteristics of the users. The physical identity verification process also offers a higher assurance verification process to verify a user's identity information through remote identity verification using the WebRTC protocol [7].

3.1 Physical Identity Acquisition

Physical identity acquisition enables users to initiate the process for verifying their real-world identities. Currently, the developed Physical Identity Acquisition module allows users to verify the following Physical Identity documents:

i. Identity Card

ii. Passport

iii. Profession Certificate (i.e., university degree certificate)

iv. Driving License

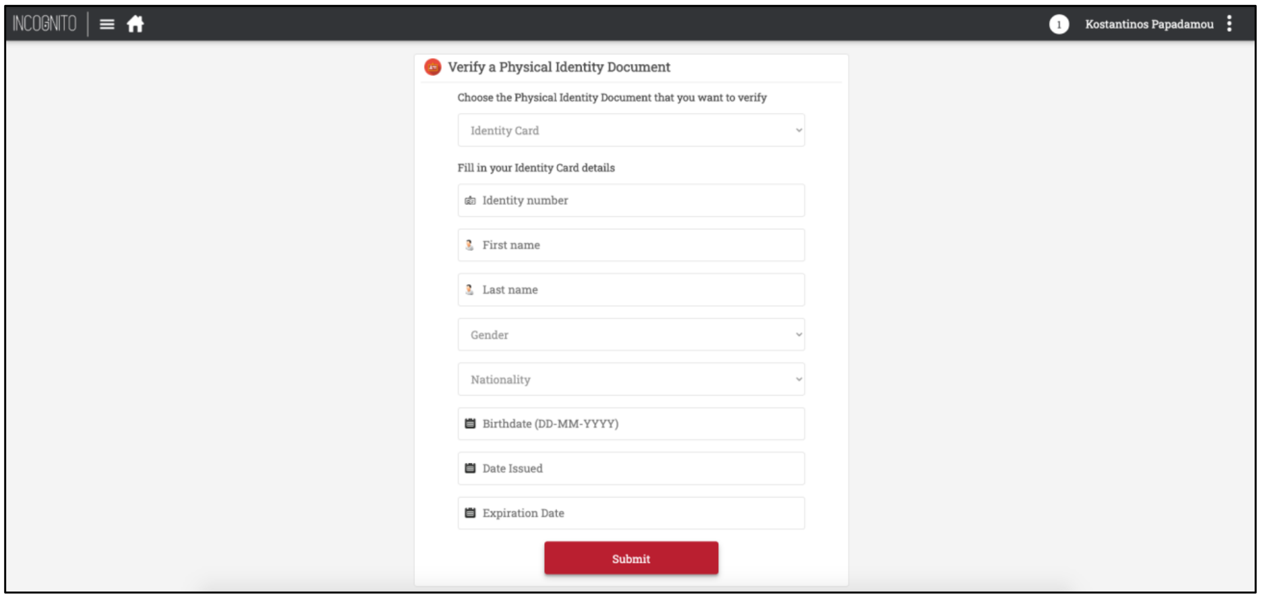
When choosing to verify a specific identity document the application requests from the user to declare the identity information included in the document like his name, surname, document number, date of birth, etc. After declaring the requested information, the user is requested to capture a photo of his whole identity document from his device’s camera. The, the user is requested to crop multiple parts from his identity document photo and also capture photos of his face from the device’s camera.

Furthermore, physical identity acquisition includes the acquirement of additional physical characteristics of the users (e.g., his location) for verification purposes that will be described in the identity verification section. Physical Identity Acquisition process allows users to declare his work and/or home address. For the verification of these addresses the application allows users to choose between two different methods that will also be described in the next section.

Moreover, the Physical Identity Acquisition module offers to the users some additional methods to declare and verify their identity attributes. First, a user is able to verify his identity remotely by presenting his/her identity document to a verifier via real-time video conference using WebRTC. The second method allows users to scan their NFC-enabled identity documents. This functionality enables the ease and secure acquirement of identity attributes without any further verification. Below we describe each one of the supported functionalities of the Physical Identity Acquisition process.

3.1.1 Initiate Physical Identity Acquisition process

Through the physical identity acquisition application, the user is able to initiate the identity acquisition and verification process for various identity documents. Initially, the user is able to choose among his identity card and passport. After the user has verified one of the two identity documents then he is able to initiate the acquisition and verification process for his profession certificate (university certificate) and driving license. As soon as the user has chosen the identity document that he wants to verify then he is requested to declare the identity information that are included in the document. This screen in the implemented web application is shown in Figure 4 below.


INCOGNITO

Kostantinos Papadamou

Verify a Physical Identity Document

Choose the Physical Identity Document that you want to verify

Identity Card

Fill in your Identity Card details

Identity number

First name

Last name

Gender

Nationality

Birthdate (DD-MM-YYYY)

Date Issued

Expiration Date

Submit

Figure 4. Declare Physical identity document information

3.1.2 Capture Physical Identity Document and User Face Pictures

The Physical Identity Acquisition module, in order to be able to generate the peer-to-peer audits and verify the declared identity information, it requests from the user to capture photos of his identity document and also of his face. Initially, the application requests from the user to capture a photo of his whole identity document. After capturing and uploading the photo of his whole identity document, the user is presented with a list of required actions requiring his to crop multiple parts from the captured identity document photo as shown in Figure 5. In the case of the identity card these parts are the document number, the name and surname, the face photo included in the identity document and the date of birth. In the case of the passport the user is requested to crop all the aforementioned and the Machine-Readable Zone (MRZ) field included in the passport also. Here, it is important to mention that all the users’ captured photos are encrypted using AES 128-bit encryption and are securely stored in a private directory in the Identity Consolidator server. Additionally, when a physical identity document is successfully verified all the related photos are deleted from the system.

3.1.2.1 Cropping parts of the Identity document photo

As mentioned above, the user is requested to crop multiple parts of his whole identity document photo. We have implemented this functionality using Jcrop, an open-source API for image cropping written in JavaScript. This API allowed us to add cropping functionality to our application. Below is a list of the parts that the user is requested to crop depending on the physical identity document that he wants to verify:

1. ***Identity Card:*** *a) Identity number; b) name and surname; c) picture of the holder as printed in the identity card; and d) date of birth.*

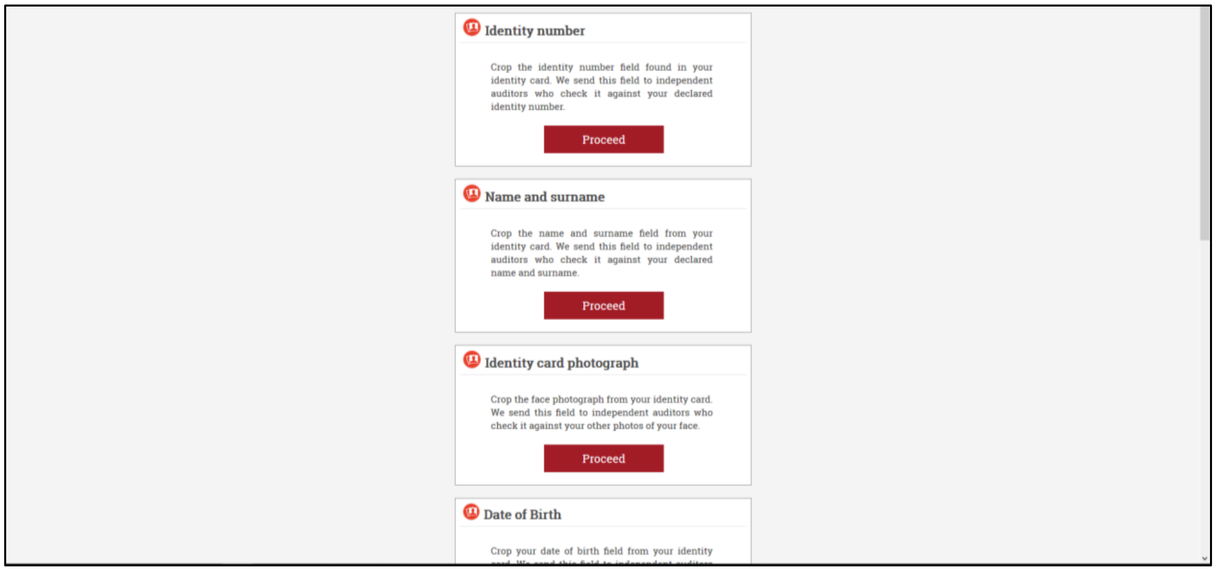
2. ***Passport:*** *a) passport number; b) name and surname; c) picture of the holder as printed in the passport; d) Machine Readable Zone (MRZ); and e) date of birth.*

3. ***Profession Certificate/University Degree Certificate:*** *a) profession/qualification; and b) name and surname.*

4. ***Driving License:*** *a) name and surname; b) picture of the holder as printed in the driving license; c) document issue date; and d) document expiration date.*

3.1.2.2 Cropping User Face photos for Verification purposes

Besides cropping parts from the captured identity photo, the user is also required to capture some more photos including a selfie for verification purposes that will be explained below in the peer-to-peer verification subsection. He is also requested to capture a photo holding his identity document, a selfie, a photo of the right side of his face and a photo of the left side of his face. These photos are requested only when the user tries to verify his identity card or passport. Furthermore, in the backend of the Physical Identity Acquisition module, as soon as the user has uploaded all the required photos for each one of the peer-to-peer audits then the appropriate audits are issued and assigned to randomly selected users of the system to ensure than no specific user will be assigned to audit all the parts of the identity of the user and this ensure to ensure the protection of the privacy of the user.


Identity number

Crop the identity number field found in your identity card. We send this field to independent auditors who check it against your declared identity number.

Proceed

Name and surname

Crop the name and surname field from your identity card. We send this field to independent auditors who check it against your declared name and surname.

Proceed

Identity card photograph

Crop the face photograph from your identity card. We send this field to independent auditors who check it against your other photos of your face.

Proceed

Date of Birth

Crop your date of birth field from your identity

Figure 5. List of required parts to crop from the captured identity document photo

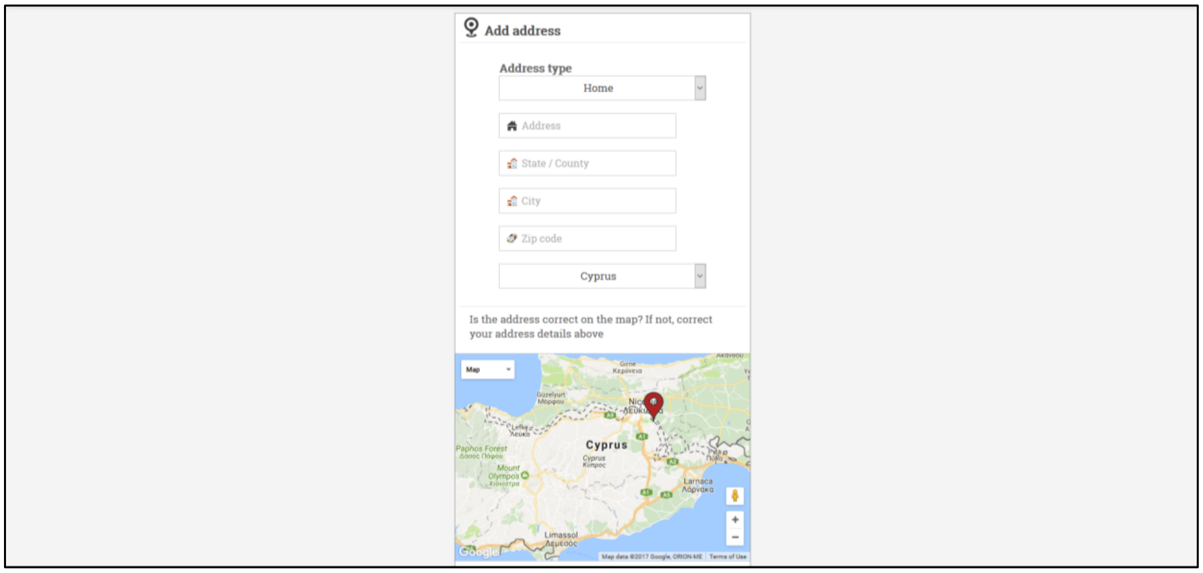
3.1.3 View and Manage Physical Identity Document Information and Uploaded Pictures

The physical identity acquisition web application that we have developed offers to the users the ability to view and manage the pictures that they have uploaded, as well as view their identity information. With this functionality, the user is able to see all the photos that he captured for one of his physical identity documents separately. At the same time, the user is able to delete any photo that he/she prefers to capture and upload a new one when for example the already uploaded one is blurred. Furthermore, when a photo is deleted, all the other photos that are related with this photo, in terms of peer-to-peer audits, are also deleted. If the user chooses to delete an Identity document photo, then all the related to this identity document photos are also deleted.

3.1.4 User Addresses' Information Acquisition

Besides, identity document acquisition and verification, the developed Physical Identity Acquisition application allows users to declare and verify their address details for both their work and home address. Using this functionality, the user has two options to declare and verify his addresses. The first is by clicking on a map where the address that he wants to declare is and the second option is to declare the details of the address while verifying on a map that the address that he wanted to declare is the correct one.

For the implementation of this functionality, we used the Google Maps JavaScript API [7], which allowed us to embed a map into this functionality. In the first option where the user is able to declare his addresses by clicking on a map, at the time when he clicks on a specific location on the map, in the background we use the Google Maps API to get the address details (street, county/state, city, country and postal code) as well as the coordinates of the selected location. In the second option we use the Google Maps API to get the coordinates of the declared address and also to show to the user the exact location on a map based on the declared address details. An example of the second option offered to the users is shown in Figure 6 below. All the methods that are used to verify the declared users' addresses will be described in Section 3.2.


Add address

Address type

Home

Address

State / County

City

Zip code

Cyprus

Is the address correct on the map? If not, correct your address details above

Figure 6. Address information acquisition by declaring its details

3.1.5 RFID-enabled Identity Document Scanning using NFC-enabled devices

Beside the identity acquisition process from standard real-world identity documents, INCOGNITO allows users to easily, quickly, and securely acquire identity attributes from RFID-enabled proofs of identity (i.e., ePassports). Using the INCOGNITO applications, users are able to scan and directly verify their identity attributes using NFC-enabled mobile devices. The implementation of this functionality is based on the findings of the IRMA project [8].

Taking advantage of the NFC protocol, this functionality enables the Physical Identity Acquisition mobile application (NFC reader) to communicate with the RFID-enabled national ID card or ePassport and securely read all the identity attributes and biometric data included in the identity document. In order the Physical Identity Acquisition application to be able to retrieve all the identity information from the identity document, the device needs to perform BAC (Basic Access Control) authentication [9] with the identity document. BAC is a mechanism used to ensure that only authorized parties can wirelessly read personal information included in RFID-enabled identity documents.

Extending ReCRED. It is important to note here that a protype of this functionality was implemented and reported in Deliverable 12 of the ReCRED project. However, in the context of task T4.1 of INCOGNITO we have extended this prototype implementation with several improvements, namely: 1) we have implemented Active Authentication that allows us to detect and defend against forged or cloned biometric identity documents, thus increasing the security and TRL level of this functionality; 2) we have implemented the acquisition of additional biometric data such as the fingerprint and face of the holder of the document; and 3) we have implemented all the required front-end and back-end functionality for the verification and storage of the acquired information. For the sake of understanding what we offer through this functionality we provide a detailed description of the RFID-enabled biometric identity document scanning using NFC below.

Active Authentication. Next, to enhance the security of our solution, we have also implemented Active Authentication (AA). Active Authentication is based on asymmetric cryptography and it is performed so that we can detect and prevent a cloned/forged biometric identity documents to be read as an authentic one. In our implementation we are performing Active Authentication in a slightly different way than the other standard biometric identity document scanners. In our case, we are performing remote Active Authentication instead of local because we want our server to be able to verify whether the data read from an RFID chip of a biometric document belongs to the genuine document. In a remote Active Authentication scenario, the server acts as the verifier while the user device (biometric document reader) acts just as a proxy that scans the data from the document and submits them to the server. In this scenario, the server generates a random challenge which is sent to the passport through the INCOGNITO mobile application. The challenge is signed by a private key residing on the biometric document's chip and the INCOGNITO mobile application reads the signed challenge, the public key of the biometric document, as well as all the other identity information and biometric data of the user and submits all of them to the backend for verification and storage. Once the server has received all the information, it uses the public key to verify the signed challenge and if everything is fine then the identity information of the user is stored securely in Identity Attributes storage.

For the implementation of this functionality, we make use of JMRTD library [10]. JMRTD is an open-source Java library that implements the Machine-Readable Travel Document (MRTD) standards as specified by the International Civil Aviation Organization (ICAO) [11]. Part of this implementation is a host side Java API, which is the one that we actually use in order to read, decode, and validate the information included in the RFID chip of the identity document. As soon as the Physical Identity Acquisition application had successfully acquired all the identity attributes of the user included in his identity document, it uses the Identity Attributes Storage API to securely store them in the Identity Attributes Storage of the Identity Acquisition and Verification platform.

Below there are there are the screenshots of the implemented functionality. We have implemented this functionality for both biometric passports and biometric identity cards. Figure 7 shows the first page of the process that a user has to follow in order to verify his biometric ID document where we explain to him the steps that he needs to follow. Next, as shown again in Figure 7 the user is requested to choose the biometric identity document type that he/she wants to scan (ePassport or national eID cards). After the user has chosen the document that he wants to scan we request from him to enter some basic information (Document number, Expiration date and date of birth) in order to perform BAC authentication and be able to read the data stored on the RFID chip of the identity document as shown in Figure 8. The next thing that the user has to do is to place his mobile device on top of his ID document in order to communicate with the installed RFID chip. After that, BAC is performed and if the provided information is correct then we send a challenge message initially received from the server to be signed by the private key included biometric document chip. Once the challenge has been signed, we also read the data stored on the document and submit them to the server (see Figure 9). The server then performs Active Authenticate verifying the signed challenge and if the signature is valid then the data are stored in the Identity Attributes Storage of the IDC.

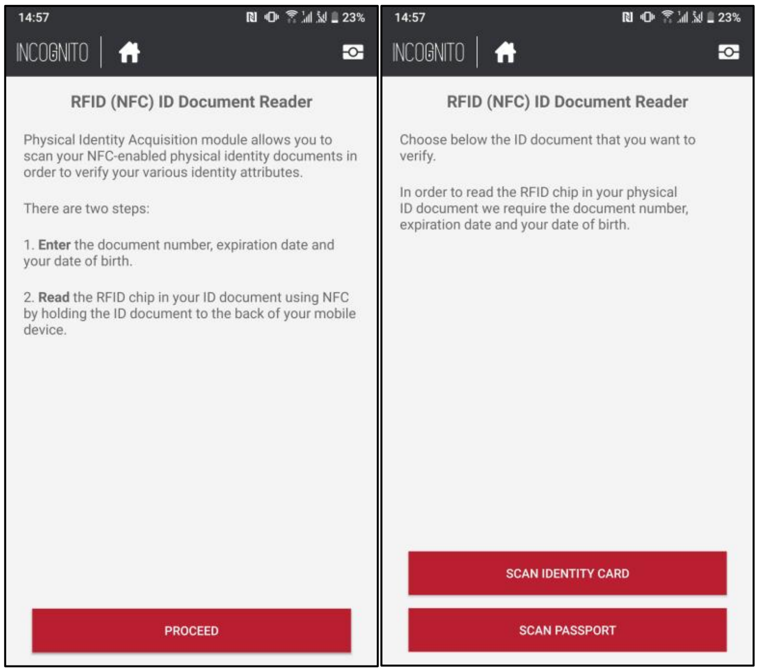
Using the implemented functionality, we are also able to acquire the biometric features (e.g., fingerprint and face of the holder) included in the biometric identity documents of the user. More precisely, if available, we are retrieving and storing the fingerprints, face, and signature of the holder as printed in his/her biometric document. This biometric data can then be leveraged to authenticate the user either as the first factor of authentication or as part of a failover authentication process in case the user loses his/her device. In this way we are also able to provide the notion of document-to-device authentication to the user. In a sense, a similar way of document-to-device authentication is also applied in almost all the airports where the user authenticates him/herself against the airports security boards before proceeding to the departing gates. Here, we note that we have also investigated other ways of incorporating biometric features included in the identity documents of the users like for example using digital signatures extracted from QR-codes and barcodes, however, we elect to implement and consider the biometric features included the physical identity documents of the users (i.e., ePassport) as it is something that users are more familiar with and it is a biometric feature that can be easily provided and used by all the intended users of the INCOGNITO platform.

3.1.5.1 Near-Field Communication (NFC) Technology

NFC technology is a form of secure contactless communication between smart devices. Such contactless communications allow users to wave the smart device over an NFC compatible device or card to send or receive information without the need to touch the devices together and with no extra steps for setting up the connection. NFC technology allows a device, known as a reader or active device, to create a radio frequency current that communicates with another NFC compatible device or a small NFC tag holding the information the reader wants. Passive devices, such as the NFC tag in smart posters, store information and communicate with the reader but do not actively read other devices. Peer-to-peer communication through two active devices is also a possibility with NFC. This allows both devices to send and receive information.

3.1.5.2 Biometric Identity Documents (ePassport/national eID card)

Biometric identity documents are also known as ePassport or digital passports or national eID cards. For example, ePassport is an improved paper and electronic passport which contains all the personal information of a person as well as his biometric information (e.g., fingerprint or biometric photograph of the owner) and can be used to authenticate his/her identity. ePassports use contactless smart card technology by having an embedded RFID chip. All the document and chip characteristics are documented by the International Civil Aviation Organization (ICAO) [11]. In order a reader device to be able to scan an ePassport and read the data stored electronically in the passport's RFID chip, it first need to authenticate against the identity document. Public key infrastructure (PKI) is used for the authentication and this makes it difficult for a malicious user to forge when all the proposed security measurements are enforced.


14:57 23%

INCOGNITO

RFID (NFC) ID Document Reader

Physical Identity Acquisition module allows you to scan your NFC-enabled physical identity documents in order to verify your various identity attributes.

There are two steps:

1. **Enter** the document number, expiration date and your date of birth.

2. **Read** the RFID chip in your ID document using NFC by holding the ID document to the back of your mobile device.

14:57 23%

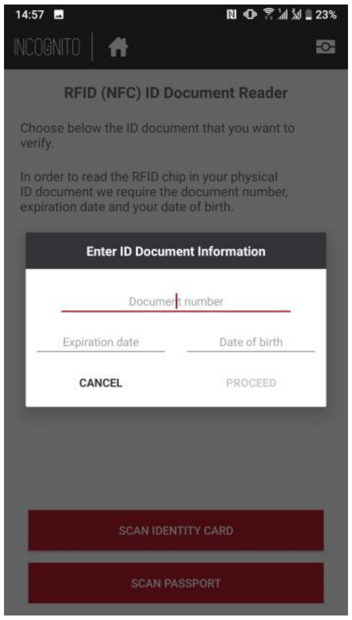
INCOGNITO

RFID (NFC) ID Document Reader

Choose below the ID document that you want to verify.

In order to read the RFID chip in your physical ID document we require the document number, expiration date and your date of birth.

Figure 7. RFID-enabled Identity document verification guidelines (left) and prompt to choose the document to be verified (right)


14:57 23%

INCOGNITO

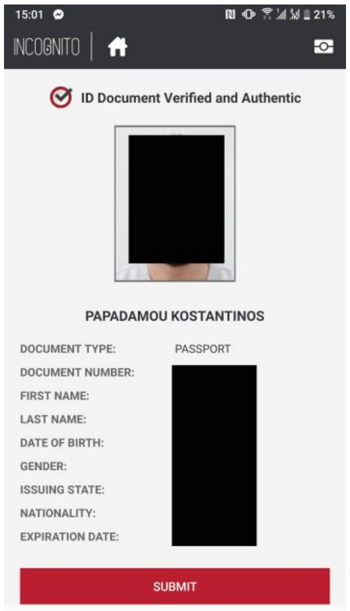
RFID (NFC) ID Document Reader

Choose below the ID document that you want to verify.

In order to read the RFID chip in your physical ID document we require the document number, expiration date and your date of birth.

Enter ID Document Information

Figure 8. Enter Identity document required information for BAC authentication



[redacted]

PAPADAMOU KOSTANTINOS

DOCUMENT TYPE: PASSPORT
DOCUMENT NUMBER:
FIRST NAME:
LAST NAME:
DATE OF BIRTH:
GENDER:
ISSUING STATE:
NATIONALITY:
EXPIRATION DATE:

[redacted]

Figure 9. Data from the biometric passport has been read. Data has been anonymized for privacy reasons.

3.1.5.3 Biometric Identity Documents' Security Mechanisms

Biometric identity documents are equipped with various security mechanisms in order to defend against various attacks. More precisely, these mechanisms are the following:

1. **Non-traceable chip characteristics:** When a reader tries to communicate and scan the RFID chip of an ePassport, the chip each time replies with a different chip number. This prevents malicious users from tracing ePassport chips.
2. **Basic Access Control (BAC):** is used to protect the communication channel between the RFID chip and the reader. The reader application needs to provide some basic information derived from the Machine-Readable Zone (MRZ) of the ID document such as date of birth, expiration date, document number, before the data stored in the RFID chip can be read.
3. **Passive Authentication (PA):** Passive authentication is used to detect modifications of the date stored on ePassports chips. The RFID chip contains a file (SOD) that stores hash values of all files stored in the chip (picture, fingerprint, etc.) and a digital signature of these hashes. The digital signature is made using a document signing key which itself is signed by a country signing key. If a file in the chip (e.g., the picture) is changed, this can be detected since the hash value is incorrect. Readers need access to all used public country keys to check whether the digital signature is generated by a trusted country.
4. **Active Authentication (AA):** Active Authentication is used to prevent cloning of ePassport RFID chips. The chip contains a private key that cannot be read or copied, but its existence can easily be proven. AA is based on a challenge response mechanism and allows the reader to verify that the data scanned from an RFID chip belongs the genuine ePassport. In our implementation AA is performed between the server and the ePassport because we want to perform remote NFC ID document scanning.
5. **Extended Access Control (EAC):** EAC offers the required functionality for checking the authenticity of both the chip and the reader. Furthermore, it uses stronger encryption than BAC and is typically

used to protect the biometric information (e.g., fingerprint, iris, etc.) stored in the RFID chips. All the European Union’s ePassports use EAC.

6. **Supplemental Access Control (SAC):** This access control scheme has been introduced later on for address BAC weaknesses.
7. **Shielding the RFID chip:** By shielding the RFID chip of the ePassport we can prevent unauthorized reading of the document. Some countries have integrated a very thin metal mesh into the passport's cover to act as a shield when the passport cover is closed.

3.2 Physical Identity Verification

The Physical Identity Verification process includes all the methods used for the verification of all the acquired identity information of the users. Among the methods used for verification are peer-to-peer audits using crowdsourcing techniques, and automatic verification methods using OCR, face detection, and face recognition techniques.

Extending ReCRED. Here we clarify that the implementation of the physical identity verification process was initiated and reported in Deliverable 12 of the ReCRED project. In the context of INCOGNITO, we have performed the following improvements: 1) we have refactored the code of all the physical identity verification processes in order to improve performance and decrease the time required for the automatic identity verification process to finish; and 2) we have implemented the automated identity verification processes using latest and more accurate machine learning algorithms for face detection and recognition[3] hence increasing the accuracy of our face detection and recognition feature to 99.38%. In this section, we provide a detailed description of all the identity verification processes that we enforce as so that the reader has all the required information to understand all the methods and processes that we employ to verify the acquired users' identity information.

Furthermore, all the implemented automatic identity verification methods have been deployed as docker containers on the backend of the Physical identity Acquisition module. Additional identity document verification methods have been implemented in the context of INCOGNITO. That is, we leverage the WebRTC protocol and we implemented a remote identity document verification process during which, a professional verifier communicates with the user that is to be verified through a video call initiated within the Identity Acquisition and Integration platform and the verifier verifies all the declared and acquired identity information of the user.

3.2.1 Peer-to-Peer Verification Using Crowdsourcing Techniques

As mentioned in the Physical identity Acquisition subsection, when a user has captured and uploaded the required photos for each identity document, in the backend of the Physical Identity Acquisition module we assign peer-to-peer audits to randomly selected users. These audits are crowdsourced to other users in order to verify that the information included in the acquired photos of another user match his declared personal identity information. Watermarking and cropping techniques are used to ensure that nobody will have access to reusable or forgeable photos of the users’ physical identity documents. Additionally, we implemented this functionality so that no more than one audit is crowdsourced to each auditor in order to prevent them to have access to more than one identity document photos of the audited user.

[3] https://github.com/ageitgey/face_recognition

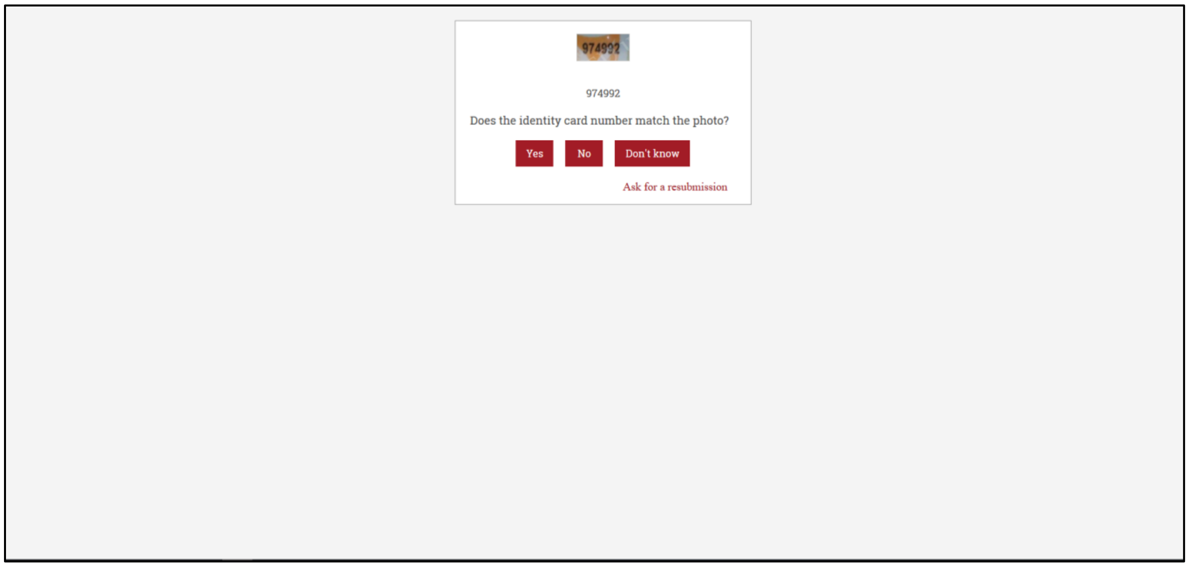
For each identity document that is supported by the Physical Identity Acquisition module we have implemented different audit types where each audit verifies a different identity attribute of the user. These types are:

1. ***Identity Card and Passport:***
   a. ***Identity Document Number.*** *The auditor is requested to verify whether the identity number shown in the cropped identity card photo matches the identity number that the audited user declared.*
   b. ***Name and Surname.*** *the auditor is requested to verify whether the name and surname shown in the cropped identity card photo matches the name and surname that the audited user declared.*
   c. ***Selfie Holding Identity Document.*** *the auditor is requested to verify whether the face of the user in the captured photo matches the identity card face photograph that the user holds in his hand.*
   d. ***Redacted Identity Document.*** *the auditor is requested to verify whether the redacted identity document in the photo is an identity card issued by the country that the audited user declared.*
   e. ***Identity Document Holder Photograph.*** *the auditor is requested to verify that all the photos include the same person and whether this person is performing a requested move in his front face photo.*
   f. ***Date of Birth.*** *the auditor is requested to verify whether the date of birth shown in the cropped identity card photo matches the date of birth that the audited user declared.*
2. **Profession Certificate**
   a. **Name and Surname.** the auditors are requested to verify whether the name and surname shown in the cropped profession certificate photo matches the name and surname that the audited user declared.
   b. **Redacted Profession Certificate.** the auditor is requested to verify whether the profession information shown in the redacted profession certificate photo matches the profession that the audited user declared.
3. **Driving License**
   a. **Name and Surname.** the auditors are requested to verify whether the name and surname shown in the cropped driving license photo matches the name and surname that the audited user declared.
   b. **Driving License Holder Photograph.** the auditors are requested to verify that all the photos include the same person and whether this person is performing a requested move in his front face photo.
   c. **Driving License Issue Date.** the auditor is requested to verify whether the date shown in the cropped driving license photo matches the driving license issue date that the audited user declared.

d. **Driving License Expiration Date.** the auditor is requested to verify whether the date shown on the cropped driving license photo matches the driving license expiration date that the audited user declared.

e. **Redacted Driving License.** the auditor is requested to verify whether the redacted driving license in the photo is a driving license issued by the country that the audited user declared.

When all the audits of a user have been successfully performed by the responsible auditors and if all of them are successful then the specific identity document is considered as verified and all the identity attributes related to this identity document are independently stored as verified into the Identity Attributes Storage. An example of crowdsourced audit is shown in Figure 10.


974992

Does the identity card number match the photo?

Figure 10. Peer-to-peer audit of the identity document number of a user

3.2.2 Automated Verification Techniques

For the verification of the acquired identity information and uploaded photos of the user, in addition to the peer-to-peer verification the Physical Identity Acquisition module uses existing techniques like OCR, face detection etc. We employ such techniques in various parts of the identity acquisition and verification process in order to automatically verify the acquired identity information and pictures of the users. More precisely, we employ the following techniques:

a. ***Optical Character Recognition (OCR):*** *is the mechanical or electronic conversion of images of typed, handwritten or printed text into machine-encoded text. We employed this technique when a user crops and uploads parts of his identity document photo in order to first detect that a text is included in each of those images and to also extract this text. In the background of our application, we extract the text from the images and we then compare it against the appropriate declare identity information to verify that they match. We have implemented this functionality using the phpOCR library, which is an open-source OCR library written in PHP that allowed us to extract text from the images that users upload.*

b. ***Face detection:*** *Face detection or facial detection is a computer technology used in a variety of applications that enables the detection of human faces in digital images. In the Physical Identity*

Acquisition module, we employ face detection to determine and verify whether a human face is included in the identity document photo and in all the photos that the user is required to capture photos of his face. We implement this functionality as a Python module, using the face_recognition Python library[4], and we deploy it on the backend of the Physical Identity Acquisition and Verification module. We prefer to have automated face detection deployed as a backend service instead of using a cloud API because the images that are process contain users' sensitive personal information that we want to protect.

c. ***Face recognition:*** *Face recognition or facial recognition systems are capable of identifying or verifying that the person included in an image is the same person included in another image or a face database. We employ face recognition in our application to determine whether the person that is included in a user's cropped identity photograph is the same person with the one included in the front face photo of the user. For the implementation of this functionality, we use the same Python library used for the implementation of the face detection functionality.*

3.2.3 Remote Identity Verification using the WebRTC Protocol

The Physical Identity Acquisition and Verification platform offers an extra layer of identity verification above the previously reported ones, which the users have the chance to request voluntarily. More precisely, we have developed another functionality that enables users to prove their identity to a professional verifier of INCOGNITO remotely through a real-time video conference. A user may ask for additional remote identity verification in order to increase the confidence score of his verified identity attributes, and thus his credibility on INCOGNITO. During remote identity verification, users are able to prove their identity through real-time video and audio presentation of their physical identity document to the professional verifier and the verifier is in place to ask further questions in order to completely verify the identity of the remote user. This functionality has been implemented using the WebRTC protocol within the Physical Identity Acquisition Web application.

For the implementation of this functionality, we use the PeerJS library [13], which is a JavaScript library that wraps the browser's WebRTC implementation and provides a complete and easy-to-use peer-to-peer connection API. In order to initiate the real-time identity presentation, the only thing that a user has to do is to enter the email of the verifier and then the appropriate user is contacted to setup the connection between the two peers. Below there are some screenshots of this functionality (see Figure 11 and Figure 12).

Web Real-Time Communication (WebRTC) Protocol: WebRTC is a collection of communication protocols and APIs that enable real-time communication including video and audio, over peer-to-peer (browsers) connections. It is an open-source project that enhances browsers, mobile applications, and Internet of Things (IoT) devices with real-time communication capabilities.

[4] https://github.com/ageitgey/face_recognition

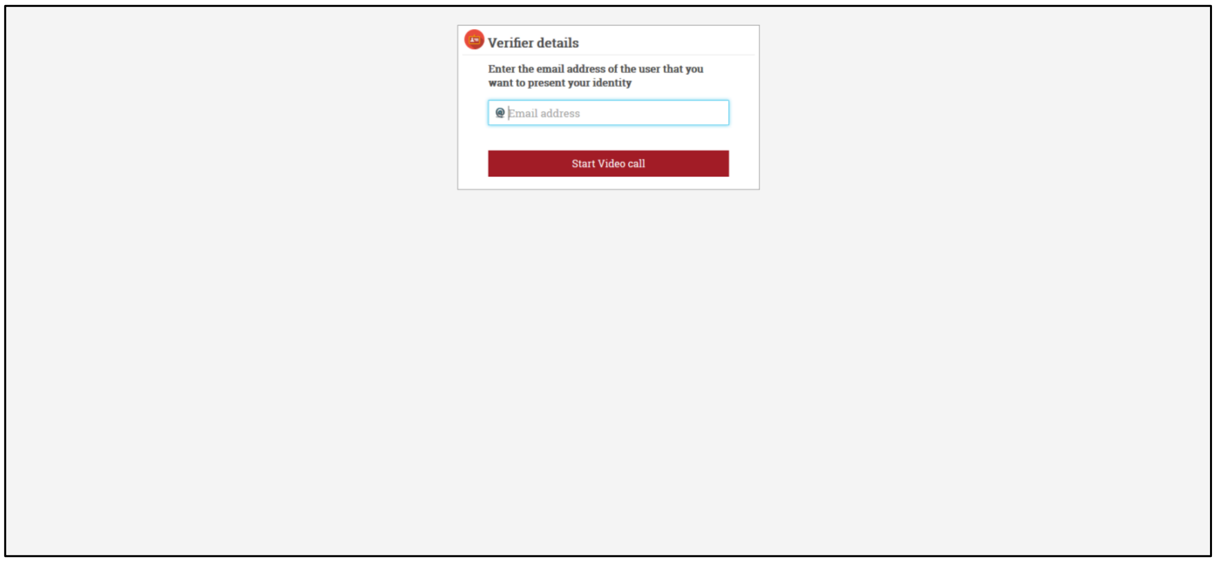
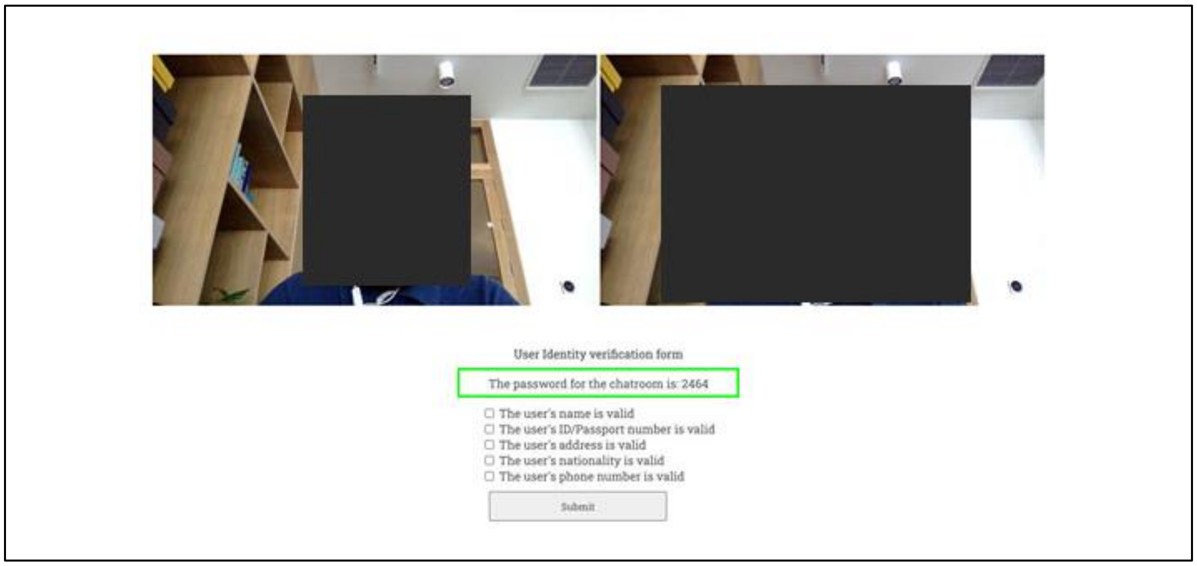

Verifier details
Enter the email address of the user that you want to present your identity
Email address
Start Video call

Figure 11. Screen where the verifier declares the email address of the user that wants to be verified using remote identity verification


[redacted]
[redacted]

User Identity verification form

The password for the chatroom is: 2464

- ☐ The user's name is valid
- ☐ The user's ID/Passport number is valid
- ☐ The user's address is valid
- ☐ The user's nationality is valid
- ☐ The user's phone number is valid

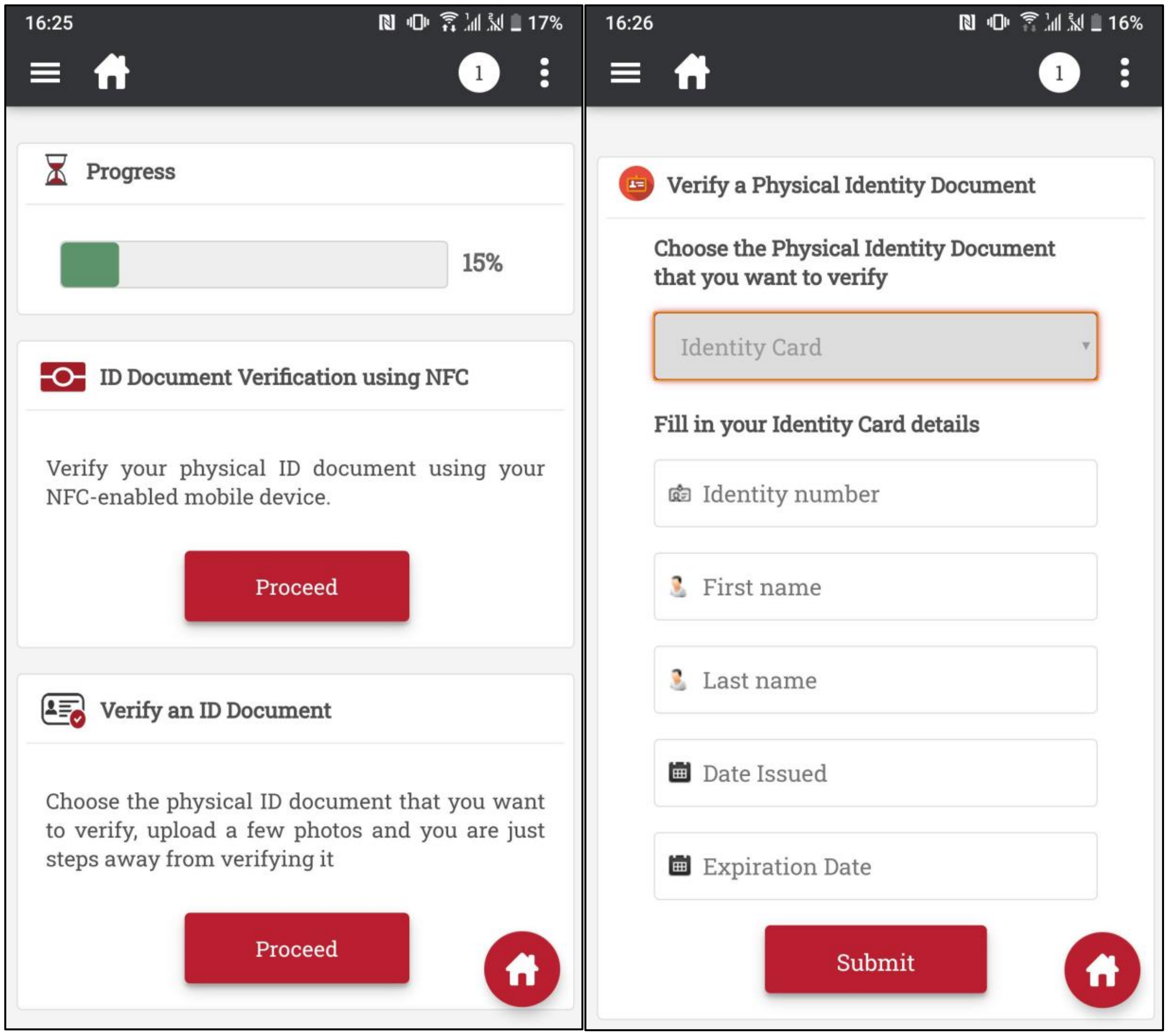
Figure 12. Remote Identity Verification using the WebRTC protocol. Faces hidden for privacy reasons

3.3 Physical Identity Acquisition Mobile Application

Besides the Physical Identity Acquisition Web application, we have also implemented an Android application using latest available technologies for Android implementation (AndroidX and RxJava), which runs on any android mobile device with Android 5+. This application supports the same functionalities as the Physical Identity Acquisition Web application. Figure 13 below shows the main page of the Physical Identity Acquisition Android application.

For the implementation of the Physical Identity Acquisition android application exploited the capabilities of the mobile devices in order to securely capture the identity information and physical characteristics of the users. Since in our application we are using all the available sensors on the devices (such as camera, GPS, etc.), a user has to authorize the application with the appropriate permissions to access theses sensors. Furthermore, in order to enable the secure capturing of photos in our application and ensure that the submitted identity documentation photos have been captured from the device’s camera we utilize the Trusted Execution Environment (TEE) offered in commodity Android devices.

All the identity information acquired from the Physical Identity Acquisition mobile application is stored to the Identity Attributes Storage using the Identity Attributes Storage API. The description for the supported functionalities of the Physical Identity Acquisition mobile application is included in the Physical Identity Acquisition and the Physical Identity Verification subsections above.


16:25
Progress
15%
ID Document Verification using NFC
Verify your physical ID document using your NFC-enabled mobile device.
Proceed
Verify an ID Document
Choose the physical ID document that you want to verify, upload a few photos and you are just steps away from verifying it
Proceed

16:26
Verify a Physical Identity Document
Choose the Physical Identity Document that you want to verify
Identity Card
Fill in your Identity Card details
Identity number
First name
Last name
Date Issued
Expiration Date
Submit

Figure 13. Screenshots of from some pages of the Physical Identity Acquisition Mobile Application

3.4 Physical Identity Acquisition Module Internal REST API

Besides the functionalities described above, the Physical Identity Acquisition module offers the Physical Identity Acquisition internal API, which is used by all the other modules of the Identity Consolidator platform when this is needed.

For the implementation of the Physical Identity Acquisition Internal API, we use Node.js Express.js framework, which is a framework for the development of Web applications and REST APIs. The REST operations that the current version of the Physical Identity Acquisition Module Internal REST API supports is described below.

3.4.1 Physical Identity Document Verification Status

Description: Provides information about the verification status for a specific identity document of a user.
Operation: GET
Request:

```
GET /physical_id_acquisition_module/api/v1/document_verification_status/{user_id}/{document_type}
Accept: application/json
```

Response:

```
200
Content-Type: application/json

{
        "user_id":"1",
        "document_type":"identity_name",
        "verified":"true",
        "verification_progress":"50%"
}
```

Description of Elements in Request URI:

Element	Description	Valid Value
user_id	User’s unique Identifier	An integer up to 5 digits
document_type	Physical Identity document type	identity, passport, profession_certificate, driving_license

Description of Elements in Response Body

Element	Description	Required	Valid Value
user_id	User’s unique Identifier	Yes	An integer up to 5 digits
document_type	The type of the physical identity document specified in the request	Yes	identity, passport, profession_certificat e, driving_license
verified	Declares whether the user has successfully verified one of his basic physical identity documents (identity card or passport)	Yes	TRUE or FALSE
verification_progress	A percentage that represents the progress of the user on successfully	Yes	Float

verifying the specified identity document

3.4.2 Identity Attribute Verification Status

Description: Provides information about the verification status for the specified identity attribute of a user.
Operation: GET
Request:

```
GET /physical_id_acquisition_module/api/v1/identity_attribute_status/{user_id}/{attribute}
Accept: application/json
```

Response:

```
200
Content-Type: application/json

{
		"user_id":"1",
		"identity_attribute":"identity_name_surname",
		"verified":"true"
}
```

Description of Elements in Request URI:

Element	Description	Valid Value
user_id	User’s unique Identifier	An integer up to 5 digits
identity_attribute	The identity attribute that you want the verification status	String

Description of Elements in Response Body

Element	Description	Required	Valid Value
user_id	User’s unique Identifier	Yes	An integer up to 5 digits
identity_attribute	The identity attribute specified in the request	Yes	String
verified	Declares whether the specified identity attribute of the given user has successfully verified or not	Yes	TRUE or FALSE

3.4.3 Store RFID-enabled Physical Identity Document Information

Description: Stores the information of the RFID-enabled Physical Identity document of a user acquired through the NFC functionality.
Operation: POST
Request:

```
POST /physical_id_acquisition_module/api/v1/store_nfc_physical_id_acquisition_document_info
Accept: application/json

{
        "user_id": 287,
        "documentType": "passport",
        "documentNumber": 12345678,
        "firsName": "John",
        "lastName": "Doe",
        "dateOfBirth": "01/01/1990",
        "gender": "male",
        "country": "England",
        "nationality": "British",
        "expirationDate": "01/01/2030",
        "document_photo": "TWFuIGlzIGRpc3Rpbmd1aX.......CBwbGVhc3VyZS4=",
        "biometric_document_pk": "SAIUBNfdaps7......AA==",
        "signed_challenge": "........."
}
```

Response:

```
200
Content-Type: application/json

{
        "result": SUCCESS
}
```

Description of Elements in Request URI:

Element	Description	Valid Value
user_id	User’s unique Identifier	An integer up to 5 digits
documentType	The type of the Physical Identity Document	String ('identity_card' or 'passport')
documentNumber	The document number in the submitted Physical Identity Document	String
firstName	The name of the user in the submitted Physical Identity Document	String
lastName	The surname of the user in the submitted Physical Identity Document	String
dateOfBirth	The date of birth in the submitted Physical Identity Document	Date
gender	The gender of the user in the submitted Physical Identity Document	String
country	The country of the user in the submitted Physical Identity Document	String
nationality	The nationality of the user included in the submitted Physical Identity Document	String

expirationDate	The expiration date of the submitted Physical Identity Document	Date
document_photo	The Base64 string of the biometric photograph included in the submitted Physical Identity Document	Base64 String
biometric_document_pk	The Public Key retrieved from the biometric identity document	String
signed_challenge	The signature of the challenge message initially sent from the server for Active Authentication	String

Description of Elements in Response Body

Element	Description	Required	Valid Value
result	The result of the storage of the submitted Physical Identity Document Information	Yes	String

4 Online Identity Acquisition Module

The Online Identity Acquisition module enables users to acquire all the identity information of a user from his/her online accounts. It is responsible for the horizontal binding of the various online identities of a user like Facebook, Google, etc., or any other Identity Provider. The account's identity information of the user can be obtained in the following two manners:

1. On the one hand, the identity acquisition process is initiated by the end-user. In this case, the Online Identity Acquisition module relies on a process defined by the OAuth2 protocol to gain access to the identity provider and retrieve the end-user account information.
2. In the second scenario, the user can grant permission to the module to automatically obtain its identity information from a service using Web crawlers that we have implemented, and which are running on the backend of the IDC.

We refer to the former as user-assisted online identity binding process and to the latter as the automatic online identity acquisition processes. Note that, the user-assisted process provides more guarantees with respect to the veracity of the collected data since the OAuth2 authentication process requires the user to authenticate with the specific Identity Provider and provide his/her explicit authorization to be able to proceed and acquire his/her account's identity information.

Furthermore, all the identity information obtained by the Online Identity Acquisition module from the multiple IdPs of supported, is stored in the Identity Attributes Storage using the Identity Attributes Storage API. In addition, the Identity Integration Module integrates the info of a user coming from different providers using as reference the algorithm described in Section 5.1. Figure 14 and Figure 15 shows the main page of the Online Identity Acquisition web and mobile applications, respectively, from which the user is able to connect his Facebook, Google, and LinkedIn account with INCOGNITO and retrieve his identity information. Also, the user is able to collect his/her identity information from other IdPs supported by INCOGNITO or he/she can enable the automatic online identity acquisition process and use our crawlers to retrieve the information of his/her online accounts.

Extending ReCRED. The implementation of this module has been initiated in the ReCRED project and reported in Deliverable 12 of that project. However, while we keep a somehow similar look and feel, we note that we have refactored the code so that we use the latest available APIs provided by Facebook, Google, and LinkedIn. At the same time, we have implemented from scratch the web crawlers that perform the automatic online identity acquisition, as python headless browser scripts using Selenium[5]. We have also deployed the crawlers as docker containers configured so that they can scale to multiple instances at the same time if this is required.

4.1 User-assisted Online Identity Acquisition process

The user-assisted online identity acquisition process is based on the user authentication flow as defined by the OAuth2 protocol which is required by all the Online Social Networks that the Online Identity Acquisition module supports (Facebook, Google, LinkedIn) in order to authenticate and authorize a third-party service (in this case the Online Identity Acquisition module) to access the user's account information. Once the chooses to use the user-assisted online identity acquisition process, he/she first has to authenticate to the respective social network. Then, the user is presented with the authorization screen where he/she is notified about the data that INCOGNITO wants to obtain access to. At this point, the user has to give explicit authorization to INCOGNITO to access the notified user's personal information. Upon the authorization of the user, the online identity acquisition module obtains an allowance to collect the user's account information and thus it proceeds to the attributes acquisition. Some attributes available in the considered IdPs (Facebook, Google, and LinkedIn) are name/surname, date of birth, location, education, occupation, etc. Upon successful acquisition, the collected attributes are then stored to the Identity Attributes Storage of the IDC through the Identity Attributes Storage REST API. Note that, all the information stored in the Identity Attributes Storage for a given IdP is always updated every time the user chooses to authenticate with each IdP.

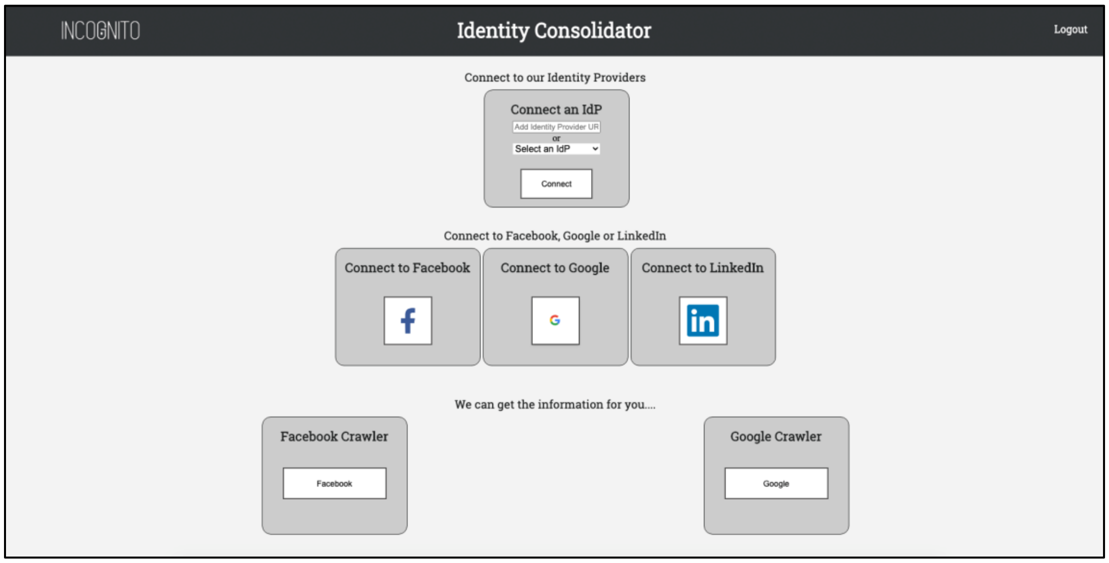


Figure 14. Main page of the Online Identity Acquisition Web application

[5] https://pypi.org/project/selenium/

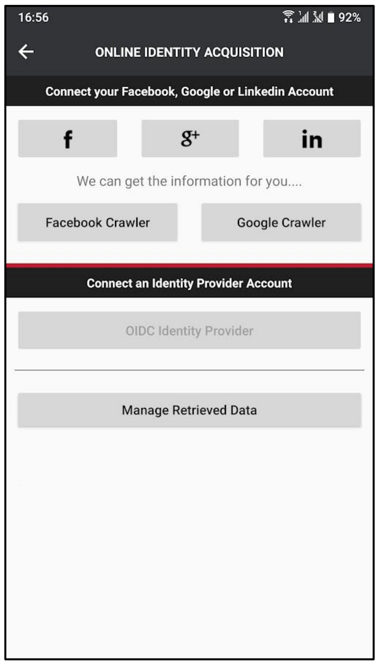

16:56
92%
ONLINE IDENTITY ACQUISITION
Connect your Facebook, Google or Linkedin Account
f
g+
in
We can get the information for you....
Facebook Crawler
Google Crawler
Connect an Identity Provider Account
OIDC Identity Provider
Manage Retrieved Data

Figure 15. Main page of the Online Identity Acquisition Mobile application

Furthermore, as indicated above, the access to the supported Identity Providers is implemented using the OAuth2 protocol, an open standard for authorization. The OAuth 2.0 enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf RFC6749 [15].

Authorization and Privacy Management: The IDC through the Online Identity Acquisition module requests from the users their authorization to access and collect information from their profiles in various IdPs. In the case of Facebook, in the authorization process, each attribute has a checkbox associated. The user should select only those attributes he is willing to share with the IDC. In the case of Google, the user cannot make such fine-grained selection. Instead, Google informs the user about the attributes to be collected and the user can grant binary permission to collect either all or to no identity attributes. Therefore, the user-assisted online identity acquisition process respects the privacy of the user since information is only collected upon the explicit authorization of the user.

4.2 Automatic Online Identity Acquisition process

The automatic online identity acquisition process is on Web crawlers that we have implemented in the context of this module for the crawling of the account information of a user from his/her profile page in various IdPs. We note that, the automatic process does not require user authentication but does require the user's consent by requiring the user to start the process once and the implemented crawlers will run in the background and collect all the personal information included in the user's IdP profile page. When the user chooses to initiate the crawling process, a pop-up message is shown to the user requesting: 1) authorization to retrieve the public information available in his/her profile in the correspondent IdP; or 2) the URL of its Facebook (or Google+) account. After this, the Online Identity Acquisition Module checks if the provided URL is correct and launches the crawling process, which retrieves the public information from the profile associated to the URL provided by the user. The collected information is stored in the

Identity Attributes Storage through the Identity Attributes Storage API. Note that, the information stored in the Identity Attributes Storage for a given IdP is updated every time the user voluntarily initiates the crawling process. To access the information available in the public profile of a user we have implemented Web crawlers using the Selenium Webdriver [16], a tool for browser automation. The Selenium Webdriver sends commands to a browser through a driver and retrieves the desired results (e.g., open a Facebook user profile page and get his username).

Authorization and Privacy Management: In the case of the automatic online identity acquisition process, the crawlers start collecting data only when the user has provided his consent to do so. In particular, the pop-up message informs the user that the information collected would be the public information available in its Facebook (or Google+) profile. Note that this information may vary across users depending on the privacy configuration set up of each user account. Therefore, the non-assisted binding process of the Online Identity Acquisition module respects the privacy of the user since information is only collected upon the explicit authorization of the user.

5 Identity Integration and Normalization

This section provides the description and the implementation details of the Identity Integration and Normalization module of the Identity Acquisition and Integration platform in general. In the context of task T4.1, we have designed and implemented a framework for integrating and normalizing, with high confidence, identity attributes obtained from the Physical Identity Acquisition and the Online Identity Acquisition modules. More precisely, this framework is responsible for aggregating and consolidating the identity information collected from the multiple IdPs of a specific user, as well as the user's real-world identity documents. The Identity Integration and Normalization module is also responsible for assigning confidence scores that indicate the veracity of the identity attributes and for labelling identity attributes based on their origin. For example, identity attributes that have been verified through remote identity verification should have a very high confidence score, while other identity attributes (e.g., email address) acquired from the Facebook profile of the user may have lower confidence score. The IDC should be able to convey to verifiers that it has acquired a specific identity attribute of a user through a named Identity Provider and let the verifier determine how much it trusts the identity information of that provider.

We have implemented the Identity Integration and Normalization framework using Python and deployed it as docker container which encapsulates a background service that runs continuously on the backend of the IDC. This background service continuously checks, for each INCOGNITO user, the Identity Attributes Storage. Once new identity attributes of a user exist in the Identity Attributes and these attributes have not already been integrated and normalized, the Identity Integration and Normalization module initiates the integration and normalization process. The integration process first needs to map the identity attributes from different IdPs referring to the same piece of information. Once the mapping has been completed, we have to define an algorithm to solve conflicts and chose one of the possible values for a given attribute. We use an algorithm that relies on the Level of Assurance (LoA) associated to different Identity Providers. In short, our algorithm, in case of a conflict between two IdPs, it selects the attribute associated with the IdP who has the higher LoA. Once all conflicts have been solved, the consolidated information for a user is stored in the Identity Attributes Storage.

5.1 Identity Integration and Normalization Algorithm

Here, we describe the algorithm implemented for the integration and normalization of the identity information collected from various Identity Providers, as well as the real-world identities of a user. The implemented algorithm is responsible for the consolidation of the collected identity information, for the resolution of any conflicts, as well as for assigning confidence scores to each normalized identity attribute. In the current implementation we consider N (6) data providers: 1) Physical Identity Acquisition module; 2) Remote Identity Verification; 3) Online Identity Providers (Facebook and Google); 4) Banks; 5) Telcos; and 6) Universities. Each Data Provider has a list of predefined attributes and a predefined LoA. The Remote Identity Verification and the Physical Identity Acquisition module have the highest LoA, and then the other Identity Providers have lower LoAs. Let’s consider a given user U for whom we have data from multiple different providers. The data integration and normalization processes are as follows:

A. If an attribute A is present only in 1 provider it passes immediately to the integrated profile of U with an assigned Confidence Score (CS) equal to the LoA of the provider, in the corresponding percentage, which provided that attribute.

B. If an attribute A is present in 2 or more providers belonging to different LoA

    i. If the attribute has been collected through remote identity verification or through the physical identity acquisition, then the value of the attribute is passed to the integrated profile of the user and the highest CS is assigned to this attribute.

    ii. If the value of the attribute matches in all Identity Providers, then the attributed is passed to the integrated profile and the CS assigned is equal to the highest LoA, in the corresponding percentage.

    iii. If the value of the attribute does not match between the different Identity Providers, the attribute associated to the provider with the highest LoA is selected and passed to the integrated profile. The CS is assigned as the Highest LoA - penalization factor, in percentage. The penalization factor (PF) increases with the number of mismatches in the attribute value but will be in any case higher to the Highest CS - 1. Example: For a given attribute we have 3 providers (P1 with LoA = 4, P2 with LoA = 3 and P3 with LoA = 2). The consolidated attribute will be the one provided by P1. If P1 mismatches with only P2 or P3 the CS will be 100 - PF1 (e.g., PF1 = 3.45 and CS = 96.55). If P1 mismatches with both P2 and P3 the CS will be 100 - PF2 (e.g., PF2 = 6.9 and CS = 93.1).

C. If an attribute A is present in 2 or more providers belonging to the same LoA and there are discrepancies, then there will be a predefined ranking between the 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 step B (iii).

In the end, the integrated profile of the user will have a final list of attributed assigned with a CS which is easily interpretable, integer numbers will indicate info coming from 1 of more providers from a given LoA without existing discrepancies. Whereas decimal numbers will reflect the LoA of the IdP which provided the consolidated attribute as well the number of mismatches existing with other providers.

6 Identity Attributes Storage

This chapter defines the implemented Identity Attributes Storage data model. The designed schema is implemented using a traditional relational database MariaDB [17], a community-developed fork of the

MySQL relational database. Note that this module serves as the central repository for the Identity Acquisition and Integration platform and the IDC in general.

Extending ReCRED. We note that the design of the Identity Attributes Storage is influenced by the design of the Identity Repository of ReCRED reported in Deliverable 12 of that project. However, in the context of task 4.1 and WP4 in general, we have re-designed the database according to the needs and requirements of INCOGNITO and we have deployed the database on the backend of the IDC as a distinct Docker container that can only be accessed by specific other Docker containers running inside the IDC, hence increasing the security of our database and protecting the data stored. We have also implemented scripts that are responsible to perform daily backups of the database schema and the data stored. Also, we will further update the Identity Attributes Storage for the needs of WP3, WP4, and WP5 and any possible updates will be reported in the respective future deliverables of the project.

In INCOGNITO, we will maintain all the collected identity information of the users, as well as other information required for the implementation of all the functionalities of the IDC, in this central repository and we will also utilize a blockchain implementation where we will store proofs of verifiable identity attributes and other information required for decentralized identity management. Details about the implementation of the decentralized identity management platform will be provided in future deliverable D4.2. All the other modules of the IDC can interact with the Identity Attributes Storage through the Identity Attributes Storage REST API to store and retrieve any information needed to develop their own functionalities. Below we provide a description of all the tables of the Identity Attributes Storage as well as their respective fields.

6.1 Notation and Conventions

In this document we use the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" as described in RFC2119[6]. Domain name examples use RFC2606[7].

6.2 User Account Table

This section outlines the fields in the central and preferred user profile structure.

Table Column	Description
Identifier	The primary key of the user in INCOGNITO.
displayName	The name that the user chooses to be displayed in all the INCOGNITO User Interfaces for his/her account.

[6] https://tools.ietf.org/html/rfc2119
[7] https://tools.ietf.org/html/rfc2606

Email	An email address at which the person may be reached.
emailConfirmUrl	An URL to verify the user email.
verifiedEmail	A boolean. True if the email has been validated.
physicalIdentityVerificationInProgress	The process of verifying (uploading photos) a specific physical identity document and which document is this none, identity, passport, driving_license, profession_certificate.
randomMove	A randomly selected number for each user. This number declares a specific random move that the user is requested to do when capturing his front face photo in physical id acquisition application.
registerDate	The date of the user registration
accountType	The type of the user in the system. In Physical ID Acquisition module 0 means administrator, 1 means a typical user and 2 means that the user is a profession auditor in physical id acquisition platform.
accountCanceled	A Boolean. True if the account has been canceled

6.3 User Physical Identities Info Table

Table Column	**Description**
Identifier	Unique identifier
primaryKey	The primary key of this entry in database.
documentType	The type of the physical documentation of the user. For instance, this filed will use values such as “Identity”, “Passport”, "Profession Certificate", "Driving License", etc.

documentNumber	The number of the user’s physical documentation. This field may be the number of his national identity or his passport number.
Gender	The gender of the user on his physical identity documentation.
dateOfBirth	The date of birth of the user on his physical identity documentation.
Nationality	The nationality of the user on his physical identity documentation.
issueDate	The issued date of the user’s physical identity documentation.
expirationDate	The expiration date of the user’s physical identity documentation.
profession	The profession of the user.
namesInfo	The primaryKey of the namesInfo field in in the talbe User Names Info, related with the Physical ID information
verificationInfo	Describes additional information about the verification process on each record (single or plural) of this Contact, which MAY be by the Identity Provider's service, or MAY be by a different service (external or peer users). Consumers SHOULD NOT assume that this data has been verified by the Identity Provider. For more information, see Verification Info details.

6.4 Identity Providers Data Table

This section outlines the fields in the normalized profile structure.

Table Column	Description
Identifier	Unique identifier
providerID	The ID which identified the provider.
primaryKey	Optional. A user ID number, usually chosen automatically by the system.

formatted	The full name, including all middle names, titles, and suffixes as appropriate, formatted for display. This is the Primary Sub-Field for this field, for the purposes of sorting and filtering.
idp_formatted	A Boolean, specifies if the user give his consent to the IdP in such field.
familyName	The family name of this Contact, or "Last Name"
idp_familyName	A Boolean, specifies if the user give his consent to the IdP in such field.
givenName	The given name of this Contact, or "First Name"
idp_givenName	A Boolean, specifies if the user give his consent to the IdP in such field.
middleName	The middle name(s) of the user.
ipd_middleName	A Boolean, specifies if the user give his consent to the IdP in such field.
Gender	The gender of the user.
idp_Gender	A Boolean, specifies if the user give his consent to the IDP in such field.
Birthday	Date of birth in YYYY-MM-DD format. Year field may be 0000 if unavailable.
idp_ Birthday	A Boolean, specifies if the user give his consent to the IDP in such field.
utcOffset	The offset from UTC of this contact’s current time zone, as of the time this response was returned.
idp_UtcOffset	A Boolean, specifies if the user give his consent to the IDP in such field.
Email	An email address at which the person may be reached.
idp_Email	A Boolean, specifies if the user give his consent to the IDP in such field.

verifiedEmail	A Boolean, indicates whether the email has been verified or not.
URL	The URL of a webpage relating to this person.
idp_URL	A Boolean, specifies if the user give his consent to the IdP in such field.
phoneNumber	A phone number at which the person may be reached.
address1	See the address field section for details.
idp_address1	A Boolean, specifies if the user give his consent to the IdP in such field.
address2	Additional address field, in case the IdP has more than one address available.
idp_address2	A Boolean, specifies if the user give his consent to the IdP in such field.
work	The workplace of the user provides to the IdP.
idp_Work	A Boolean, specifies if the user give his consent to the IdP in such field.
education	The education place of the user provides to the IdP.
idp_Education	A Boolean, specifies if the user give his consent to the IdP in such field.
limitedData	A boolean value, true if social login was able to retrieve only limited public data from the user’s profile (for example, because the login session has expired, or the user logged out from their account).
Trusted	A boolean reflecting whether the user trusts the Identity Consolidator to store the actual data.
updated	The current timestamp when the data is saved for the first time or is updated.

6.5 User Names Info Table

This table contains the components of the contact's real name.

Table Column	Description
identifier	Unique identifier
primaryKey	The primary key of this entry in database.
formatted	The full name, including all middle names, titles, and suffixes as appropriate, formatted for display. This is the Primary Sub-Field for this field, for the purposes of sorting and filtering.
CS_formatted	The confidence score calculated for the formatted field.
LoA_formateed	The level of assurance of the consolidated data for the formatted field.
familyName	The family name of this user, or "Last Name"
CS_familyName	The confidence score calculated for the familyName field.
LoA_familyName	The level of assurance of the consolidated data for the familyName field.
givenName	The given name of the user, or "First Name"
CS_givenName	The confidence score calculated for the givenName field.
LoA_givenName	The level of assurance of the consolidated data for the givenName field.
middleName	The middle name(s) of the user
CS_middleName	The confidence score calculated for the middleName field.
LoA_middleName	The level of assurance of the consolidated data for the middleName field.
honorificPrefix	The honorific prefix(es) of the user

CS_honorificPrefix	The confidence score calculated for the honorificPrefix field.
LoA_honorificPrefix	The level of assurance of the consolidated data for the honorificPrefix field.
honorificSuffix	The honorifix suffix(es) of the user
CS_honorificSuffix	The confidence score calculated for the honorificSuffix field.
LoA_honorificSuffix	The level of assurance of the consolidated data for the honorificSuffix field.
updated	The most recent date the details of this Person were updated (i.e., the modified date of this entry). The value MUST be a valid Date. If this Person has never been modified since its initial creation, the value MUST be the same as the value of published.
verificationInfo	Describes additional information about the verification process on each record (single or plural) of this Contact, which MAY be by the Identity Provider's service, or MAY be by a different service (external or peer users). Consumers SHOULD NOT assume that this data has been verified by the Identity Provider. For more information, see Verification Info details.
moduleID	An identifier of the module who store the data in that table, in order to differentiate if the information has been provided by the end-user, the Physical Identity Acquisition Module or the Identity Integration Module or other

6.6 User Phones Info Table

This table contains Phone numbers of a user.

Table Column	Description
identifier	Unique identifier
primaryKey	The primary key of this entry in database.
phoneNumber	A phone number at which the person may be reached.
CS_phoneNumber	The confidence score calculated for the phoneNumber field.
LoA_phoneNumber	The level of assurance of the consolidated data for the phoneNumber field.
type	The type of phone, with Canonical Values home, work, other. The list of Canonical Values could be extended in a future release.

updated	The most recent date the details of this Person were updated (i.e., the modified date of this entry). The value MUST be a valid Date. If this Person has never been modified since its initial creation, the value MUST be the same as the value of published.
verificationInfo	Describes additional information about the verification process on each record (single or plural) of this Contact, which MAY be by the Identity Provider's service, or MAY be by a different service (external or peer users). Consumers SHOULD NOT assume that this data has been verified by the Identity Provider. For more information, see Verification Info details.
moduleID	An identifier of the module who store the data in that table, in order to differentiate if the information has been provided by the end-user, the Physical Identity Acquisition Module or the Identity Integration Module or other

6.7 User Addresses Info Table

This table stores the components of a physical mailing address.

Table Column	Description
identifier	Unique identifier
primaryKey	The primary key of this entry in database.
formatted	The full mailing address formatted for display or use with a mailing label. This field MAY contain newlines. This is the Primary Sub-Field for this field, for the purposes of sorting and filtering.
CS_formatted	The confidence score calculated for the formatted field.
LoA_formateed	The level of assurance of the consolidated data for the formatted field.
streetAddress	The full street address component, which may include house number, street name, PO BOX, and multi-line extended street address information. This field MAY contain newlines.
CS_streetAddress	The confidence score calculated for the streetAddress field.

LoA_streetAddress	The level of assurance of the consolidated data for the streetAddress field.
city	The city or locality component.
CS_city	The confidence score calculated for the city field.
LoA_city	The level of assurance of the consolidated data for the city field.
region	The state or region component.
CS_region	The confidence score calculated for the region field.
LoA_region	The level of assurance of the consolidated data for the region field.
postalCode	The zipcode or postal code component.
CS_postalCode	The confidence score calculated for the postalCode field.
LoA_postalCode	The level of assurance of the consolidated data for the postalCode field.
country	The country name component.
CS_country	The confidence score calculated for the country field.
LoA_country	The level of assurance of the consolidated data for the country field.
type	The type of address, with Canonical Values home, work, other. The list of Canonical Values could be extended in a future release.
latitude	The latitude value of the address
longtitude	The longtitude value of the address

updated	The most recent date the details of this Person were updated (i.e. the modified date of this entry). The value MUST be a valid Date. If this Person has never been modified since its initial creation, the value MUST be the same as the value of published.
verificationInfo	Describes additional information about the verification process on each record (single or plural) of this Contact, which MAY be by the Identity Provider's service, or MAY be by a different service (external or peer users). Consumers SHOULD NOT assume that this data has been verified by the Identity Provider. For more information, see VerificationInfo details.
moduleID	An identifier of the module who store the data in that table, in order to differentiate if the information has been provided by the end-user, the Physical Identity Acquisition Module or the Identity Integration and Normalization Module or other.

6.8 MediaItem Table

This table stores information about the media items (video, image, sound) of the user.

Table Column	Description
identifier	Unique identifier
primaryKey	The primary key of this entry in database.
documentCategor y	Denotes the type of the physical document that the photo is, for example identity means that it is a photo related the identity card of the user. The valid categories are passport, identity, negative, driving_license, utility_bill, profession_certificate.
documentCategor yMediaType	Denotes the type of the picture: whole, redacted, name_surname, document_number, nationality, photo_in_document, date_of_birth, machine_readable_zone, face_photo, right_side_photo, left_side_photo, holding_id_document, profession, issued_date, address, other.
url	Location corresponding to this media item.

title	The title of the media item. This information is supplied by the user.
description	Description of the content of the media item. This information is supplied by the user.
thumbnailUrl	URL to a thumbnail cover of the media item.
mediaMimeType	String identifying the mime-types of media item in the media item.
updated	The most recent date the details of this Person were updated (i.e., the modified date of this entry). The value MUST be a valid Date. If this Person has never been modified since its initial creation, the value MUST be the same as the value of published.
verificationInfo	Describes additional information about the verification process on each record. For more information, see Verification Info details.

6.9 Verification Table

Describes additional information about the verification process on each record of the table.

Table Column	Description
identifier	Unique identifier
primaryKey	The primary key of this entry in database.
status	The status of the verification process. Identity Provider SHOULD support at least the following values: initiated, onProcess, verified.
startDate	The date this verification process has been initiated for the last time. The verification process could be a repeated one (periodically or by request). This information refers to the last time that this process has been initiated. This value SHOULD be a valid xs:date if possible, but MAY be an unformatted string, since it is recognized that this field is often presented as free-text.
endDate	The date this verification process has been initiated for the last time. The verification process could be a repeated one (periodically or by request). This information refers to the last time that this process has been initiated. This value SHOULD be a valid

	xs:date if possible, but MAY be an unformatted string, since it is recognized that this field is often presented as free-text.
verifiedBy	The name of the verification agent (e.g., company, user, Identity Provider or other organization). This field MUST have a non-empty value for each value returned.
description	A textual description of the verification process. This field MAY contain newlines.

6.10 Audits Info Table

Table Column	Description
identifier	Unique identifier
primaryKey	The primary key of this entry in database.
url	This is a random alphanumeric generated when the audit is assigned and will be used for implementation purposes. It MUST be unique for each entry in this table.
verifier_id	The identifier of the user who performs the peer-to-peer audit (verifier).
audited_id	The identifier of the user that is being verified
auditType	The type of the audit. This field can take integer values each one indicating a different type of audit.
document_type	The type of the identity document that the peer-to-peer audits corresponds.
status	The status result of the audit. The value of this field can take any of the following values: 'assigned', 'positive', 'negative', and 'undetermined'.
comment	The comment that the verifier can leave while performing the peer-to-peer audit.

name_check	The result of the peer-to-peer audit for the name attribute of the user. IT can be NULL, True or False.
surname_check	The result of the peer-to-peer audit for the surname attribute of the user. IT can be NULL, True or False.
nationality_check	The result of the peer-to-peer audit for the nationality attribute of the user. IT can be NULL, True or False.
document_number_check	The result of the peer-to-peer audit for the document number attribute of the user. IT can be NULL, True or False.
profession_check	The result of the peer-to-peer audit for the profession attribute of the user. IT can be NULL, True or False.
birthdate_check	The result of the peer-to-peer audit for the date of birth attribute of the user. IT can be NULL, True or False.
notifyResult	Denotes whether the user should be notified about the result of the specific peer-to-peer verification. True if the user should be notified, False if the user has already been notified.
dateAssigned	The datetime that the audit has been assigned to the auditor.

6.11 Cryptographic Credentials Table

Table Column	Description
identifier	Unique identifier
primaryKey	The primary key of this entry in database.
cryptoCredential	The issued cryptographic credential.
issueDatetime	The date and time that the cryptographic credential has been issued. This information is filled automatically when the credentials is stored.
expirationDatetime	The date and time that the cryptographic credential will be expired. This information is provided by the user, but it can be filled as NULL as soon as the user doesn't specify it.

Description	The description of the cryptographic credential. This information describes the identity attributes used to issue this cryptographic credential.
included_attributes	The attributes included in the credential
id_provider	A foreign key from table “Providers”, to indicate which Id Provider provides the source financial data

6.12 Service Providers Users Table

Table Column	Description
Identifier	The foreign key to the User Account identifier.
primaryKey	The primary key of this entry in database.
account_address	The URL of the 3rd party account
account_username	User’s account name in the 3rd party service. User’s real name could be “John Doe”, but on Facebook he would login as “joedeen”. A user MAY have more account usernames, which would result in separate rows in the table.
approved	A Boolean indicates if the IDC has confirmed the validity of the Service Provider.
description	The description to the user account at this service provider (e.g.: e-banking account, shopping account)
riskLevel	The risk level of the account, can be: Low, Medium or High.

6.13 Identity Providers Users Table

Table Column	Description
Identifier	The foreign key to the User Account identifier.
primaryKey	The primary key of this entry in database.

providerID	The foreign key to the Service Provider entry.
riskLevel	The risk level of the account, can be: Low, Medium or High.

6.14 Identity Providers Metadata Table

Table Column	Description
primaryKey	The primary key of this entry in database.
providerID	Unique identifier for the Identity Provider. Each Provider returned MUST include a non-empty id value. This identifier MUST be unique across this user's entire set of Contacts, but MAY not be unique across multiple users' data. It MUST be a stable ID that does not change when the same contact is returned in subsequent requests.
providerName	The name of the Provider ID
description	The description of the provider
levelAssurance	The level of assurance of the given Identity Provider
supportedAttributes	List all the attributes that the Identity Provider supports.
endpoints	List the endpoints that the Identity Provider provides.

6.15 User Physical Identity Documents Cropped Regions Table

Table Column	Description
Identifier	The primary key of the user in the database.
primaryKey	The primary key of this entry in database.
xAxis	The start points of the cropped document at x axis.

yAxis	The start points of the cropped document at y axis.
width	The width of the cropped document.
height	The height of the cropped document.

6.16 User Relationships Info Table

This table stores the relations between users in the system. When the type of the relationship is set to audit the relation between the two users is in terms of peer-to-peer audits. Each time a crowdsourced audit for a user is assigned to another user in the system a relationship is stored between the two users so that in the future the selected auditor will not be assigned an audit for the same user.

Table Column	Description
primaryKey	The primary key of this entry in database.
Identifier1	The identifier of the user 1, related with user 2.
Identifier2	The identifier of the user 2, related with user 1.
relationshipType	The type of the relationship between the two users: 'audited', 'friendship', 'following', 'transaction'.

7 Identity Attributes Storage REST API

This section provides the description of the Identity Attributes Storage REST API. This API defines a language- and platform-neutral protocol for Consumers to request, store and modify information of a user identity profile in the Identity Attributes Storage of the IDC. This information contains identity attributes of the user and proofs of account ownership as well as other personal information that are available in the Identity Attributes Storage.

Extending ReCRED. As with some other modules described in this deliverable, the implementation of the Identity Attributes Storage REST API was initiated in the ReCRED project and reported in Deliverable 12 of ReCRED. However, since in INCOGNITO we have different/additional requirements and we have created a different database schema (see Section 6), there was a need to re-implement this API. Hence, in the context of task T4.1 of INCOGNITO, we have re-factored the codebase of this REST API so that it uses the INCOGNITO database schema (Identity Attributes Storage), while also working across improving the maintainability, performance, deployability and scalability of the API. More specifically, we have performed several performance optimizations including the caching information wherever this is feasible so that the API respond faster to certain API calls that mostly utilized by the clients of this API. We have deployed the Identity Attributes Storage REST API as a docker container that can be easily scaled to multiple instances of the API based on the demand. Last, we have made the required modifications so that the codebase of this API is able to incorporate future updates regarding the support for performing operations on the INCOGNITO blockchain that will be developed in the context of task T4.2, hence acting as a node of this blockchain for certain operations. Below we provide a detailed documentation of the Identity Attributes Storage REST API, irrespectively of whether some of information are part of the initial version of the API since it is important for the reader to have the complete documentation of the API in order to understand and utilize it.

7.1 REST Services

INCOGNITO's Identity Attributes Storage REST API consist of collections of REST-accessible resources that can be accessed and modified using the basic set of HTTP request methods (GET, PUT, PATCH, POST, and DELETE). Such services have to be used by internal modules of the Identity Consolidator to access identity data hosted and managed by the IDC.

In general, for all REST Services:

- HTTP GET method is used to retrieve representations of the current state of any given resource
- PUT and PATCH are used to modify the current state
- DELETE is used to delete the resource
- POST is used to either create new resources or to perform other types of operations that do not fit within the scope of the other core HTTP methods

When a client application needs to communicate with the Identity Attributes Storage REST API via an intermediary that restricts the use of certain standard and extension HTTP Methods (e.g., PUT, DELETE, and PATCH), the client COULD utilize the "X-HTTP-Method-Override" HTTP Request Header mechanism in a POST request. The "X-HTTP-Method-Override" header MUST NOT be used to send HTTP GET requests.

Responses to all requests specify an appropriate HTTP Status Code indicating the status of the response. All REST Services share a common, basic URI Structure that MAY be extended on a case-by-case basis. This common structure helps to ensure that all interactions remain as consistent as possible across multiple REST Services while allowing individual service-specific and implementation specific behaviors to be supported.

7.2 HTTP Request & Response Header Fields

INCOGNITO services MAY use the request and response headers as defined at OData specification[8]. In particular, it is highly recommended to support the following header elements:

- **DataServiceVersion**: Clients MAY use the DataServiceVersion header on a request to specify the version of the protocol used to generate the request.
- **Content-Type**: The format of an individual request or response body MUST be specified in the Content-Type header of the request or response.
- **Accept**: As specified in RFC2616[9], the client MAY specify the set of accepted formats through the use of the Accept Header.

[8] https://www.odata.org/
[9] https://tools.ietf.org/html/rfc2616

- **If-Match**: A client MAY include an If-Match header in a request to GET, PUT, MERGE, PATCH or DELETE an entity or entity property, or to invoke an action bound to an entity. The value of the If-Match request header MUST be an ETag value previously retrieved for the entity. If specified, the request MUST only be invoked if the specified value matches the current ETag value of the entity. If the value does not match the current ETag value of the entity for a Data Modification or Action request, the service MUST respond with ‘412 Precondition Failed’ and MUST ensure that no data is modified as a result of the request.
- **ETag**: A request that returns an individual entity MAY include an ETag header. The value specified in the ETag header may be specified in the If-Match (or If-None-Match) header of a subsequent Data Modification or Action request in order to apply optimistic concurrency in updating, deleting, or invoking the action bound to, the entity.

7.3 Request Operations

An INCOGNITO service MUST support Create, Update, and Delete operations for all of the identity information and elements that it exposes. A successfully completed Data Modification request must not violate the integrity of the data. A client may request whether content be returned from a CREATE, UPDATE, or DELETE request, or the invocation of an Action, by specifying the Prefer Header.

7.3.1 Creation of Information

To CREATE an entity in a collection, send a POST request to that collection’s URL. The POST body MUST contain a single valid entity representation. Upon successful completion, the response MUST contain either a Location header that contains the edit URL of the created entity, or '201 Created', or '204 No Content' if the request included a Prefer header with a value of 'return-no-content'.

7.3.2 Conditional Requests

A client MAY include an ETag value in the if-match or if-none-match request header of a Data Modification or Action request. If specified, the operation MUST only be invoked if the if-match or if-none-match condition is satisfied. The ETag value specified in the if-match or if-none-match request header may be obtained from an ETag header of a request for an individual entity or may be included for an individual entry in a format-specific manner.

Note that the Entity Tag specified in the response is generally specific to the actual payload included in the response. If a REST service supports multiple representation formats for a single resource, such as offering multiple data format options or modified views of the resource tailored to the authentication credentials included in the request, the Entity Tag can vary for each specific response, regardless of whether the actual state of the resource on the server has changed. Therefore, for any single resource, multiple Entity Tags can potentially represent the current state of the resource.

7.3.3 Full Vs Partial Modifications

Some UPDATE requests support two types of update: replace and merge. The client chooses which to execute by which HTTP verb it sends in the request. The current state of a resource may be modified in part or in full using either the PATCH or PUT HTTP methods, respectively. A PUT request indicates a replacement update. Given a URI that represents a resource, the current state of that resource can be modified in full by sending an HTTP PUT request to the URI. The payload of the PUT request is considered to be a replacement for the identified resource, although the server is free to determine exactly how the resource is to be modified.

Alternatively, the application can use a PATCH request to perform a partial modification of the resource. A PATCH or MERGE indicates a differential update. The service MUST replace exactly those property values that are specified in the request body. Missing properties, including dynamic properties, MUST NOT be altered. The semantics of PATCH are defined in RFC 5789[10]. The service MUST be compliant with that definition. Support for the PATCH method to perform partial modifications of resource is optional. Assuming the change is successful, the server would respond with an appropriate code status, and MAY include the updated representation of the resource.

7.3.4 Delete Information

To DELETE an existing entity, send a DELETE request to that entity’s edit URL. The request body SHOULD be empty. Upon successful deletion, the response MUST be '204 No Content'.

7.4 Query Invocations

All requests to the Identity Attributes Storage are made as HTTP GET operations on a URL deriving from the specified Base URL. Consumers MAY append additional path information and/or query string parameters to the Base URL as part of the request. Additionally, authentication information MAY be sent via POST data or additional HTTP headers in the request. Responses are returned in the body of the HTTP response, formatted as JSON or XML, depending on what is requested. Response and error codes SHOULD be transmitted via the HTTP status code of the response (if possible) and SHOULD also be specified in the body of the response. Since the API endpoint is dynamic (and does not serve static content), Consumers MUST NOT interpret any cache headers in the response as having meaning because the same URL request might return a different response upon subsequent invocation.

7.4.1 Authentication & Authorization

The data returned by any endpoint of the Identity Attributes Storage API, MAY contain public data, or it MAY contain private data. If the data returned is public, no authentication or authorization is required. However, in most of the cases the data returned are private sensitive personal identity information, hence an authentication and authorization is required. Typically, this is done by the clients of the Identity Attributes Storage API obtaining either Direct Authorization, Basic Authorization with a username and password or Secured Authentication with an access token obtained out-of-band by the user and given to the Consumer to present as part of the request.

7.4.1.1 Direct Authorization

The Identity Attributes Storage API provides support for Direct Authorization through the support of HTTP Basic Access Authentication RFC2617[11], and also supports additional Direct Authorization mechanisms, if the clients choose so.

7.4.1.2 Authorization Methods Supported

The clients that want to access private data MAY choose not to support either Direct Authorization, depending on their security requirements, but they MUST support either OAuth or HTTP Basic Authorization if they require any Authorization. When accessing an endpoint of the Identity Attributes Storage API, if sufficient authorization credentials are not provided, the API returns an HTTP 401 Unauthorized response, and provides the available Authorization mechanisms available by including

[10] https://tools.ietf.org/html/rfc5789
[11] http://portablecontacts.net/draft-spec.html#RFC2617

WWW-Authenticate headers in the response for each type of Authorization method supported (as defined in RFC2616[12]). In this way, the clients will then be able to recognize that the API endpoint they are calling is a protected resource and initiate the proper Authorization process needed to obtain the appropriate credentials.

7.4.2 Query Parameters

The Identity Attributes Storage API defines a standard set of operations that can be used to filter, sort, and paginate response results. The operations are specified by adding query parameter to the Base URL, either in the query string or as HTTP POST data. In particular, we use the OData[13] protocol, which is an application-level protocol for interacting with data via RESTful interfaces. The protocol supports the description of data models and the editing and querying of data according to those models. Specifically, OData provides the following:

- **Metadata:** A machine-readable description of the data model exposed by a particular data provider.
  http://consolidator-incognito.socialcomputing.eu/INCOGNITO/StorageAPI/open/$metadata
- **Data:** Sets of data entities and the relationships between them.
- **Querying:** Requesting that the service performs a set of filtering and other transformations to its data, then returns the results.
- **Editing:** Creating, updating, and deleting data.
- **Operations:** Invoking custom logic
- **Vocabularies:** Attaching custom semantics

The path of the URL specifies the target of the request. Additional query operators, such as filter, sort, page, and projection operations are specified through query options.

7.4.3 Data Filtering

Filtering is used to limit the request results to data that match a set of given criteria. The $filter system query option restricts the set of items returned. INCOGNITO uses OData protocol and therefore the following operations and query filters shown in the two tables below are supported.

Operator	Description	Example
Logical Operators		
Eq	Equal	/User_Accounts?$filter=Identifier **eq** 1
Ne	Not equal	/User_Accounts?$filter=Identifier **ne** 1
Gt	Greater than	/User_Accounts?$filter=Identifier **gt** 10
Ge	Greater than or equal	/User_Accounts?$filter=Identifier **ge** 10
Lt	Less than	/User_Accounts?$filter=Identifier **lt** 10

[12] http://portablecontacts.net/draft-spec.html#RFC2616
[13] http://www.odata.org/

Le	Less than or equal	/User_Accounts?$filter=Identifier **le** 10
And	Logical and	/User_Accounts?$filter=Identifier **le** 10 and Identifier **gt** 3
Or	Logical or	/User_Accounts?$filter=Identifier **le** 3 and Identifier **gt** 10
Not	Logical negation	/User_Accounts?$filter=**not** endswith(Email,'com')

Function	**Example**
String Functions	
bool substringof(string po, string p1)	/User_Accounts?$filter=substringof('hills', Email) **eq** true
bool endswith(string p0, string p1)	/User_Accounts?$filter=endswith(Email,'com') **eq** true
bool startswith(string p0, string p1)	/User_Accounts?$filter= startswith(Email, 'hills') **eq** true
string tolower(string p0)	/User_Accounts?$filter=tolower(DisplayName) **eq** ‘louisa’

7.4.3.1 Sorting of Results

Sorting allows requests to specify the order in which data records are returned. The $orderby System Query option specifies the order in which items are returned from the service. The value of the $orderby System Query option contains a comma-separated list of expressions whose primitive result values are used to sort the items. A special case of such an expression is a property path terminating on a primitive property. A type cast using the qualified entity type name is required to order by a property defined on a derived type. The expression can include the suffix ASC for ascending or desc for descending, separated from the property name by one or more spaces. If ASC or DESC is not specified, the service MUST order by the specified property in ascending order. Items are sorted by the result values of the first expression, and then items with the same value for the first expression are sorted by the result value of the second expression, and so on.

7.4.3.2 Presentation of Results

Presentation controls the format, makeup, and delivery mechanism for returning the requested result set. The $select system query option requests that the service return only the properties, dynamic properties, actions and functions explicitly requested by the client. The service returns the specified content, if available, along with any available expanded navigation properties, and MAY return additional information. The value of the $select query option is a comma-separated list of properties, qualified action names, qualified function names, the star operator (*), or the star operator prefixed with the namespace or alias of the schema in order to specify all operations defined in the schema. If the $select query option is not specified, the service returns the full set of properties or a default set of properties. The default set of properties MUST include all key properties. If the service returns less than the full set of properties, either because the client specified a select or because the service returned a subset of properties in the absence of a select, the context URL MUST reflect the set of selected properties and expanded navigation properties.

The $format system query option specifies the media type of the response. The $format query option, if present in a request, MUST take precedence over the value(s) specified in the Accept request header. The value of the $format query option is a valid internet media type, optionally including parameters.

7.5 Response Format

The structure of the response object returned from a successful request MUST follows the OData specification. OData defines semantics around the following request and response headers. Additional headers may be specified, but have no unique semantics defined in OData. In addition, the data are returned in both XML and JSON formats.

7.6 Error Codes

The Identity Attributes Storage API MUST returns a response code with every response. Response codes are numeric and conform to existing HTTP response codes where possible, as defined in OData. In addition to the response code, the API SHOULD also provide a human-readable reason that explains the reason for the response code. This message SHOULD be intelligible to developers, but MAY be unsuitable for display to end-users. The clients are responsible to provide their own appropriate error message to users when encountering an error response.

7.7 Structure

Each field is defined as either a *Singular Field*, in which case there MUST NOT be more than one instance of that field per contact, or as a *Plural Field*, in which case any number of instances of that field MAY be present per user profile. Identity information is formatted using labeled attributes with either structured or unstructured string data. Each attribute value consists of one of the following types:

- *Simple:* A single string attribute which MAY specify a REQUIRED data format or allow any string. A simple field MAY contain Canonical Values specified, in which case Identity Providers SHOULD try to conform to those values if appropriate, but MAY provide alternate string values to represent additional values.
- *Boolean*: A special case of a Simple Field with two legal values: true and false. Values are case-sensitive.
- *Complex*: A multi-value attribute that contains any combination of other attributes. Complex attributes are defined by listing the child attributes and their types. For most Complex Fields, the value sub-field contains the Major Value of that field (i.e., the primary piece of contact information described by that field), and the other fields provide additional meta-data.

7.7.1 Singular Fields

Some examples of singular fields are the following:

- *id*: Unique identifier for the table record in the database
- displayName: The name of the user, suitable for display to end-users. Each Contact returned MUST include a non-empty displayName value. The name SHOULD be the full name of the Contact being described if known (e.g., John Doe), but MAY be a username or handle, if that is all that is

available. The value provided SHOULD be the primary textual label by which this Contact is normally displayed by the Identity Provider when presenting it to end-users.
- *updated*: The most recent date the details of this Contact were updated (i.e. the modified date of this entry). The value MUST be a valid xd:dateTime (e.g. 2008-01-23T04:56:22Z). If this Contact has never been modified since its initial creation, the value MUST be the same as the value of published. Note the updatedSince Query Parameter can be used to select only contacts whose updated value is equal to or more recent than a given xs:dateTime. This enables Consumers to repeatedly access a user's data and only request newly added or updated contacts since the last access time.
- *birthdate*: The date of birth of the user. The value MUST be a valid xs:date (e.g. 1975-02-14. The year value MAY be set to 0000 when the age of the Contact is private, or the year is not available.
- *gender*: The gender of the user. Identity Providers SHOULD return one of the following Canonical Values, if appropriate: male, female, or undisclosed, and MAY return a different value if it is not covered by one of these Canonical Values.

7.7.2 Plural Fields

Within this specification, a "plural-field" is a property whose value consists of zero or more alternative choices represented as individual elements (phones, emails, etc.). Unless specified otherwise, all Plural Fields have the same three standard sub-fields:

- *value*: The primary value of this field, e.g., the actual e-mail address, phone number, or URL. When specifying a sortBy field in the Query Parameters for a Plural Field, the default meaning is to sort based on this value sub-field. Each non-empty Plural Field value MUST contain at least the value sub-field, but all other sub-fields are optional.
- *type*: The type of field for this instance, usually used to label the preferred function of the given contact information. Unless otherwise specified, this string value specifies Canonical Values of work, home, and other.
- *primary*: A Boolean value indicating whether this instance of the Plural Field is the primary or preferred value of for this field, e.g., the preferred mailing address or primary e-mail address. Identity Providers MUST NOT mark more than one instance of the same Plural Field as primary="true", and MAY choose not to mark any fields as primary, if this information is not available. For efficiency, Identity Providers SHOULD NOT mark all non-primary fields with primary="false” but should instead omit this sub-field for all non-primary instances.

When returning Plural Fields, the Identity Attributes Storage API SHOULD canonicalize the returned value, if this is appropriate (e.g., e-mail addresses and URLs). The API MAY also return the same value more than once with different types (e.g., the same e-mail address may used for work and home), but SHOULD NOT return the same (type, value) combination more than once per Plural Field, as this complicates processing by the Consumer. Some examples of such plural fields are:

- *emails*: E-mail address for this Contact. The value SHOULD be canonicalized by the Identity Provider, e.g., john.doe@example.com instead of JDOE@EXAMPLE.COM.
- *urls*: URL of a web page relating to this Contact. The value SHOULD be canonicalized by the API. In addition to the standard Canonical Values for type, this field also defines the additional Canonical Values blog and profile.
- *phoneNumbers*: Phone number for the user. No canonical value is assumed here. In addition to the standard Canonical Values for type, this field also defines the additional Canonical Values mobile, fax, and pager.

- *addresses*: A physical mailing address for the user.
- *accounts*: An online account held by the user.

7.7.3 Name Element

The components of a user's real name. The API MAY return just the full name as a single string in the formatted sub-field, or they MAY return just the individual component fields using the other sub-fields, or they MAY return both. If both variants are returned, they SHOULD be describing the same name, with the formatted name indicating how the component fields should be combined. The following attributes are included in the Name element:

- *formatted*: The full name, including all middle names, titles, and suffixes as appropriate, formatted for display (e.g., Mr. John Doe). This is the Primary Sub-Field for this field, for the purposes of sorting and filtering.
- *familyName*: The family name of the user.
- *givenName*: The given name of the User, or "First Name".
- *middleName*: The middle name(s) of the user or the "Father's Name".
- *honorificPrefix*: The honorific prefix(es) of the user, or "Title".
- *honorificSuffix*: The honorifix suffix(es) of the user, or "Suffix".
- *updated*: The most recent date the details of this Person were updated (i.e., the modified date of this entry). The value MUST be a valid Date. If this Person has never been modified since its initial creation, the value MUST be the same as the value of published.

7.7.4 address Element

The components of a physical mailing address. The Identity Attributes Storage API MAY return just the full address as a single string in the formatted sub-field, or they MAY return just the individual component fields using the other sub-fields, or they MAY return both. If both variants are returned, they SHOULD be describing the same address, with the formatted address indicating how the component fields should be combined. The following attributes are included in the address element:

- *formatted*: The full mailing address, formatted for display or use with a mailing label. This field MAY contain newlines. This is the Primary Sub-Field for this field, for the purposes of sorting and filtering.
- *streetAddress*: The full street address component, which may include house number, street name, PO BOX, and multi-line extended street address information. This field MAY contain newlines.
- *city*: The city or locality component.
- *region*: The state or region component.
- *postalCode*: The zipcode or postal code component.
- *country*: The country name component.
- *updated*: The most recent date the details of this Person were updated (i.e. the modified date of this entry). The value MUST be a valid Date. If this Person has never been modified since its initial creation, the value MUST be the same as the value of published.

7.7.5 account Element

Describes an online identity/account held by a user, which MAY be another Identity Provider or a Service Provider. The clients SHOULD NOT assume that this account has been verified by INCOGNITO to actually belong to the user. For each account, the domain is the top-most authoritative domain for this account, e.g., www.amazon.com or www.google.com, and MUST be non-empty. Each account must also contain a non-empty value for either username or user_id, and MAY contain both, in which case the two values MUST be for the same account. These accounts can be used to determine if a user on one service is also known to be the same person on a different service, to facilitate connecting to people the user already knows on different services. The account element contains the following attributes:

- *domain*: The top-most authoritative domain for this account, e.g., "amazon.com". This is the Primary Sub-Field for this field, for the purposes of sorting and filtering.
- *username*: An alphanumeric user nickname, usually chosen by the user, e.g., "test1245".
- *userid*: A user ID number, usually chosen automatically, and usually numeric but sometimes alphanumeric.
- *updated*: The most recent date the details of this Person were updated (i.e., the modified date of this entry). The value MUST be a valid Date. If this Person has never been modified since its initial creation, the value MUST be the same as the value of published.
- *Trusted*: Boolean reflecting whether the user trusts the Identity Consolidator to store the actual data.

7.7.6 verificationInfo Element

Describes additional information about the verification status on each identity attribute (single or plural) of a user, which MAY be by the IDC, or MAY be by a different service (external or peer users). Consumers SHOULD NOT assume that this data has been verified by the Identity Provider. The verificationInfo element includes the following attributes:

- *status*: The status of the verification process. Identity Provider SHOULD support at leat the following values: initiated, onProcess, verified.
- *startDate*: The date this verification process has been initiated for the last time. The verification process could be a repeated one (periodically or by request). This information refers to the last time that this process has been initiated. This value SHOULD be a valid xs:date if possible, but MAY be an unformatted string, since it is recognized that this field is often presented as free-text.
- *endDate*: The date this verification process has been completed for the last time. The verification process could be a repeated one (periodically or by request). This information refers to the last time that this process has been completed. This value SHOULD be a valid xs:date if possible, but MAY be an unformatted string, since it is recognized that this field is often presented as free-text.
- *verifiedBy*: The name of the verification agent (e.g., company, user, Identity Provider or other organization). This field MUST have a non-empty value for each value returned.
- *description*: A textual description of the verification process. This field MAY contain newlines.

7.7.7 Other Elements

We note that similar elements exist for all the respective tables of the Identity Attributes Storage (see Section 6). Although in this section we omit some of them for brevity, information from any table in the database can be retrieved in the same exact way as documented above and each returned element will include most of the attributes of that database table in JSON format.

8 Identity Consolidator (IDC) 3rd Party REST API

The Identity Consolidator 3rd Party REST API that the Identity Consolidator platform exposes is used by external from the IDC entities (e.g., Service Providers, Identity Providers, etc.) for communication purposes with the IDC. In general, the IDC 3rd Party REST API is used by any INCOGNITO component or online service who wants to interact with the Identity Consolidator platform.

Extending ReCRED. We note that the role of the 3rd Party REST API and its name is influenced by the identical REST API reported in Deliverable 12 of the ReCRED project. However, in the context of task T4.1 of INCOGNITO we have implemented this API from scratch using the Node.js Express.js framework[14], which is a framework for implementing fast and scalable REST APIs. Leveraging the latest advancements in virtualization techniques, we have also deployed this REST API as a Docker container and it is configured so that multiple instances can be deployed in case this is needed. The current version of the 3rd Party API supports two operations which we describe below, and we plan to implement additional operations based on the future requirements of tasks T4.2 and T4.3, as well as the requirements of WP3 (e.g., adding operations for the creation and backup of Idemix cryptographic credentials) and WP5.

8.1 High-Level Operations

The high-level operations that the current version of the IDC 3rd Party REST API supports can be classifier to the following categories:

1. Service Providers REST Operations
    a. Transfer trust between Service Providers
2. User Account Management
    a. Delete User IDC account

We note, that as the implementation of the INCOGNITO framework and the IDC platform progresses, we will continuously update the IDC 3rd Party REST API with additional endpoints based on the required functionality that is needed to be implemented. Hence, an updated version of the IDC 3rd Party REST API with additional endpoints will be provided in deliverable D4.2.

8.2 Transfer Trust Between Service Providers

Description: This REST operation is used to transfer trust among Service Providers. For example, it can be used to transfer trust from eBay to Amazon.
Operation: POST
Request:

```
POST /api/v1/transfer_trust/
Accept: application/json
```

Request Body:

```
{
		"user_id":721,
		"source_domain":"test1.com",
		"destination_domain":"test2.com"
}
```

[14] https://expressjs.com/

Response:
```
200
Content-Type: application/json

{
		"user_id":721,
		"source_domain":"test1.com",
		"destination_domain":"test2.com",
		"attribute_description":"trust",
		"trust_value":"37",
		"transfer_datetime":"02/02/17 15:38"
}
```

Description of Elements in Request Body:

Element	Description	Valid Value
user_id	User’s unique Identifier	Integer
source_domain	The domain that is the holder of the trust value that we want to transfer	URL
destination_domain	The domain that will receive the trust value	URL

Description of Elements in Response Body

Element	Description	Required	Valid Value
user_id	User’s unique Identifier	Yes	Integer
source_domain	The domain that is the holder of the trust value that we want to transfer	Yes	URL
destination_domain	The domain that will receive the trust value	Yes	URL
attribute_description	The type of the identity attribute that we transfer. In this case this should be set to "trust"	Yes	"trust"
trust_value	The value of the trust of the user in the source domain	Yes	String
transfer_datetime	The date and time that the trust transferred.	Yes	DATETIME

8.3 Delete User Identity Consolidator Account

Description: Using this REST API call the user is able to request the deletion of his INCOGNITO account and all its stored information and credential using his/her mobile device.
Operation: POST
Request:

```
POST /api/v1/delete_account/
Accept: application/json
```

Request Body:

```
{
        "email_address": "incognitouser@gmail.com",
}
```

Response:

```
200
Content-Type: application/json

{
        "result": "SUCCESS"
}
```

Description of Elements in Request Body:

Element	Description	Valid Value
email_address	The email address of the user who wants to delete his INCOGNITO account	Email Address

Description of Elements in Response Body

Element	Description	Required	Valid Value
result	The result of the account deletion	Yes	String (SUCCESS or FAILED)

9 Conclusion

The Identity Acquisition and Integration platform and the IDC platform in general, are the main components of the INCOGNITO architecture. These components and their corresponding implemented Web and mobile applications play a vital role in both the "Bot-or-not/Fake News Dissemination" and the "Online Media Content Sharing" use cases of the INCOGNITO platform. The purpose of this deliverable, "Identity Acquisition and Integration Platform", was to describe the design and implementation of the Identity Acquisition and Integration platform and the IDC in general since the Identity Acquisition platform is part of the IDC. More precisely, in this deliverable we provided a detailed description of the implementation of the full identity acquisition Web and mobile application, which enable the acquisition and seamless integration of the multiple soft proofs of identities of a user, both the real-world identities (e.g., national ID cards, utility bills, ePassports, eID cards, university degree certificate, etc.), as well as all the online accounts of the user.

Furthermore, this deliverable also provides a description of the schema of the Identity Attributes Storage that is responsible to store and secure the collected identity information of the users as well as to hold other information that are required by several functionalities of the Identity Acquisition and Integration platform. In addition, we provide the description of the Identity Attributes Storage REST API, which allows all the modules of the IDC to interact with the users' identity information stored in the Identity Attributes Storage. Last, we also provide a description of the IDC 3rd Party REST API, which allows all the external entities of INCOGNITO to interact with the IDC.

We note that, the developed solutions and the research activities performed in the context of this deliverable fully tackle the objectives of Task 4.1. The developed Identity Acquisition and Integration platform, as well as the Identity Consolidator, which both developed in the context of this deliverable are considered as some of the key achievements of the project outcomes as they enable the acquisition, verification, integration, and normalization of the multiple soft proofs of identities of the user. At the same time, the work and research activities described in this document will be the basis for the implementation of Task 4.2 and Task 4.3. More precisely, through our continuous research activities we will build upon the modules described in this deliverable in order to tackle the objectives of Task 4.2 and implement a decentralized identity management platform leveraging the blockchain technology. This work will be documented in D4.2. In addition, the Identity Consolidator will be enhanced with a user-friendly consent management platform as part of the work of Task 4.3 and this will be part of the research activities that will be described in the deliverable D4.3.

Last, we believe that the achieved outcomes of Task 4.1 and solutions developed in the context of this task provide to the user the notion of document-to-service authentication as it allows them to consolidator all their multiple soft proofs of identities (both physical and online) under a single platform and later use their INCOGNITO account and verified identity to securely authenticate and access various Service Providers by just providing their verified identity attributes without the need to maintain and provide a username or a password.

Finally, we note that the reported scientific outcomes have also been disseminated to the scientific community in the following project publication: "Killing the Password and Preserving Privacy With Device-Centric and Attribute-Based Authentication" [20]. In addition, the key achievements and the reported scientific outcomes of this deliverable have also been disseminated to the scientific community and the general public in the following talks of the project's coordinator:

- 6th Meeting of the European Security and Defense College. Brussels. November 22, 2019
- Mastering Enterprise Management and INCOGNITO. Piraeus. November 24, 2019
- Critical Infrastructure Security and Resilience (CISaR) Workshop. Norway. January 30-31, 2020
- Secure Internet Day 2020, IoT: Challenges & Risks. Cyprus. February 11, 2020
- 7th Information Security Conference. Athens. February 19, 2020
- European Association of Biometrics Conference 2020 (EAB-RPC 2020). Virtual Event. September 16, 2020

For more information about the project's outcomes and dissemination activities, please visit our project's website[15].

[15] https://incognito.socialcomputing.eu/news-events/

10 References

[1] "OpenID Connect," 2020. [Online]. Available: https://openid.net/connect/.

[2] I. E. T. F. (IETF), "RF 6749 - The OAuth 2.0 Authorization Framework," [Online]. Available: https://tools.ietf.org/html/rfc6749.

[3] T. P. Group, "PHP: Hypertext Processor," 2020. [Online]. Available: https://www.php.net/.

[4] A. S. Foundation, "Apache HTTP Server Project," 2020. [Online]. Available: https://httpd.apache.org/.

[5] T. Otwell, "Laravel PHP Framework," 2020. [Online]. Available: https://laravel.com/.

[6] Z. Říha, "An Overview of Electronic Passport Security Features," *IFIP Summer School on the Future of Identity in the Information Society,* pp. 151-159, 2008.

[7] G. Developers, "Google Maps JavaScript API," 2020. [Online]. Available: https://developers.google.com/maps/documentation/javascript/overview.

[8] P. b. D. Foundation, "IRMA. Put a digital passport on your own mobile," 2020. [Online]. Available: https://www.irmacard.org/.

[9] Wikipedia, "Basic Access Control," 2020. [Online]. Available: https://en.wikipedia.org/wiki/Basic_access_control.

[10] J. Team, "JMRTD Project," 2020. [Online]. Available: https://jmrtd.org/about.shtml.

[11] I. C. A. Organization, "International Civil Aviation Organization (ICAO)," [Online]. Available: https://www.icao.int/Pages/default.aspx.

[12] G. Developers, "WebRTC - Real-time communication for the Web," 2020. [Online]. Available: https://webrtc.org/.

[13] P. Team, "PeerJS - Simple Peer-to-Peer with WebRTC," 2020. [Online]. Available: https://peerjs.com/.

[14] A. S. R. A. P. B. a. t. S. F. T. Josh Lockhart, "Slim - A Microframework for PHP," 2020. [Online]. Available: https://www.slimframework.com/.

[15] I. E. T. F. (IETF), "The OAuth 2.0 Authorization Framework," 2012. [Online]. Available: https://tools.ietf.org/html/rfc6749.

[16] S. F. Conservancy, "SeleniumHQ Browser Automation," 2020. [Online]. Available: https://www.selenium.dev/.

[17] M. Foundation, "MariaDB," [Online]. Available: https://mariadb.org/about/.

[18] P. Contacts, "Portable Contacts Schema Definition," 2020. [Online]. Available: http://portablecontacts.net/draft-spec.html#schema.

[19] Wikipedia, "OpenSocial Specification," 2020. [Online]. Available: https://en.wikipedia.org/wiki/OpenSocial.

[20] S. Z. B. C. S. T. G. G. A. C. A. R. C. P. G. B. S. G. C. X. M. S. Kostantinos Papadamou, "Killing the Password and Preserving Privacy with Device-Centric and Attribute-based Authentication," *IEEE Transactions on Information Forensics and Security,* vol. 15, pp. 2183-2193, 2020.

[21] I. Square, "Near-Field Communication (NFC)," 2017. [Online]. Available: http://nearfieldcommunication.org/.

[22] OData, "OData v2.0 – The Protocol for REST APIs," 2020. [Online]. Available: https://www.odata.org/documentation/odata-version-2-0/uri-conventions/.