



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



WP5 – Attribute-Based Access Control

Deliverable D5.2 “Full Design and Prototype of the ABAC Infrastructure”

Editor(s): Giuseppe Bianchi (CNIT), Alberto Caponi (CNIT), Claudio Pisa (CNIT)

Author(s): Giuseppe Bianchi (CNIT), Alberto Caponi (CNIT), Claudio Pisa (CNIT), Tooska Dargahi (CNIT), Emanuele Altomare (CNIT), Vangelis Bagiatis (UPCOM), Lefteris Fanos (UPCOM), Savvas Zannettou (CUT), Kwnstantinos Papadamou (CUT), Sotirios Chatzis (CUT), Charalampos Partaourides (CUT), Panagiotis Nikitopoulos (UPRC), Christos Lyvas (UPRC), Christoforos Dadoyan (UPRC), Eleni Veroni (UPRC), Christos Xenakis (UPRC), George Gugulea (CSGN), Bogdan Chifor (CSGN), Alberto Carp (CSGN), Mihai Togan (CSGN), Evangelos Kotsifakos (WEDIA), Bharadwaj Pulugundla (VERI), Steven Gevers (VERI)

Dissemination Level: PU









Nature: O

Version: 1.1

ReCRED Project Profile

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

Partners

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

Document History

Version	Date	Author	Remarks
0.1	18/01/2017	CNIT	TOC proposal
0.2	26/01/2017	CNIT	TOC Update
0.3	14/02/2017	CNIT	Final ToC
0.3	Feb 2017	CNIT, CSGN	FIDO/P-ABAC integration
0.4	Mar 2017	CNIT, VERI	OpenAM/P-ABAC integration
0.5	Mar 2017	CNIT, CSGN	FIWARE specification and integration
0.6	24/03/2017	CNIT	Contributions integration
0.7	27/03/2017	UPRC	TEE contribution integration
0.8	27/03/2017	CNIT	First draft release
0.9	30/03/2017	CNIT, CUT, WEDIA	Review of the deliverable
1.0	31/03/2017	CNIT	Final Release
1.1	31/07/2017	CNIT	Changes to reopened deliverable

Executive Summary

This document is part of the WP5 (Attribute-Based Access Control) of the ReCRED project. The purpose of this deliverable is to report the definition and the description of the final specification and design of the Attribute-Based Access Control (ABAC) architecture. The design of the final architecture starts from the initial design provided in the previous Deliverable D5.1 (Specification and Initial Design of the ABAC Infrastructure) which was bootstrapped by an initial investigation on the state-of-the-art of ABAC platforms to be integrated and deployed in the ReCRED framework.

The main purpose of this document is to provide a detailed description of the design and the implementation of components that provide to the ReCRED platform capabilities to enable Attribute Based Access Control (ABAC). First, a general description of the final architectural design of the ABAC architecture and protocols is provided. Next, following the whole architecture design, a detailed description of the implementation of the functionalities supported by each component is provided as well as representative screenshots from the applications that have been prototyped to offer those functionalities to the administrators and end-users. Last, a description of the usage of the ABAC capabilities in different ReCRED use-cases is provided as well as a discussion on security and privacy of components and protocols of the ABAC architecture.

We improved the initial architectural design discussed in D5.1 by focusing on the integration of ABAC components at the core of the architecture (Idemix, U-Prove, ABE) by means of a common interface provided by the FiWARE API specification. Moreover, the usage of the TEE on the user’s device, permits to execute cryptographic operations in a secure environment that prevents secret key leakages. One of the most relevant objectives reached by the integration of ABAC architecture in the ReCRED framework is the design and implementation of FIDO and OpenID connect. This integration will enable the anonymous credential system to be used in commercial authentication solutions already adopting such authentication mechanisms.

The document is organized as follows:

Chapter 2 provides a description of the Access Control mechanisms and motivations for using Attribute Based Access Control. The discussions of the state of the art for Attribute Based Access Control technologies and architectures, the definition of attributes and policies to be used for verification, final prototyping of P-ABAC components and the integration between them are discussed in Chapter 3. The application scenarios of the ReCRED project and the benefit of using ABAC on them are reported in Chapter 4. To conclude the deliverable, security and privacy considerations on the technologies discussed in the document are reported in Chapter 5, while our final general conclusions are included in Chapter 6.

Table of Contents

Executive Summary.....	4
List of Figures	7
1 Introduction	9
1.1 Attribute-Based Access Control	9
1.2 Integration of ABAC in ReCRED	11
2 P-ABAC Architecture	13
2.1 P-ABAC Architectural Overview and Relation to the ReCRED Architecture	13
2.1.1 ABAC Components	13
2.1.2 ReCRED Components Mapping to the ABAC Architecture	15
2.2 Detailed P-ABAC Architectural Description	17
2.2.1 Privacy Preserving Attribute-Based Credential Systems.....	17
2.2.2 Common Interfaces and Protocols.....	41
3 P-ABAC Module Implementation and Mapping to P-ABAC components.....	60
3.1 P-ABAC Components Implementation	60
3.1.1 Credential Management Daemon.....	60
3.1.2 U-Prove Implementation.....	68
3.1.3 Trusted Execution Environment.....	75
3.1.4 Consent Management.....	78
3.1.5 De-anonymization Risk Assessment.....	86
3.1.6 P-ABAC and FIDO Integration.....	90
3.1.7 Credential Backup	94
3.1.8 OpenAM-based P-ABAC	100
3.1.9 IRMA-FIWARE Integration	102
3.1.10 Attributes and Policies for P-ABAC	104
3.1.11 ABE-Based P-ABAC Solution for Wi-Fi.....	109
3.2 P-ABAC Components Mapping	118
4 Application Scenarios	120
4.1 Support to Financial Services.....	120
4.1.1 Before ABAC	120
4.1.2 After ABAC.....	121
4.2 Campus Wi-Fi and Campus-Restricted Web Services.....	122
4.2.1 Before ABAC	123
4.2.2 After ABAC.....	123

4.2.3	Towards Privacy-Preserving ABAC	126
4.3	Age Verification	127
4.3.1	Before ABAC	128
4.3.2	After ABAC.....	128
4.4	ISIC Student Discounts.....	129
4.4.1	Before ABAC	129
4.4.2	After ABAC.....	129
5	Privacy and Security Considerations.....	131
5.1	Attacks and Privacy Issues in ABE and ABAC.....	131
5.2	Lack of Revocation.....	131
5.3	Key Abuse Attack for KP-ABE.....	131
5.4	Key Escrow.....	131
5.5	Attribute Hiding Attack in ABAC.....	131
5.6	Revelation of Access Policy and Attributes to Untrusted Servers.....	131
5.7	Revelation of User's Identity in Multi Authority Scheme	132
5.8	Mitigations to Enhance Security in ABE and ABAC systems.....	132
5.9	Privacy considerations of Idemix and U-Prove.....	132
5.9.1	Threat Model.....	132
5.9.2	Comparison of Privacy features	132
5.9.3	Revocation.....	133
5.10	OpenAM P-ABAC Security Considerations	134
5.11	FIDO Extensions for ABAC and Anonymous Credentials	134
5.12	Security Considerations for the IRMA-FIWARE integration	134
5.13	Privacy and Security Considerations in Idemix and U-Prove implementation.....	134
5.14	Security Analysis of the ABE-Based P-ABAC solution for Wi-Fi	135
6	Conclusions / Future Work	136
7	References	137

List of Figures

Figure 1 ReCRED ABAC functional architecture with elements involved	14
Figure 2 ABAC components view of the ReCRED Architecture.....	15
Figure 3 Idemix Issuance Protocol rounds between Issuer and Recipient components	19
Figure 4 Idemix Proving Protocol rounds between Prover and Verifier components.....	23
Figure 5 U-Prove Issuance protocol	31
Figure 6 U-Prove Presentation Protocol	35
Figure 7 "Using IRMA is easy" - from the IRMA Project [9]	54
Figure 8 IRMA data flow (verification)	55
Figure 9 IRMA for Mobile Devices Architecture	55
Figure 10 IRMA Issuance sequence diagram	56
Figure 11 IRMA Verification Sequence Diagram.....	57
Figure 12: Consent Management Design.....	58
Figure 13. IRMA Issuance Protocol	61
Figure 14. U-Prove client-server architecture.....	62
Figure 15. Credential Management module submodules and interactions	63
Figure 16. List of available credential ready to be issued.....	65
Figure 17. Credential Issuing: Link of the User Device to the API server by means of QR Code	65
Figure 18. ReCRED Wallet Application	66
Figure 19. ReCRED Wallet Application: issuance consent display	67
Figure 20. ReCRED Wallet application: Credential list and details	68
Figure 21 The class diagram for the U-Prove engine	69
Figure 22 The issuance protocol from U-Prove	73
Figure 23: Idemix Issue protocol workflow implemented in TEE by means of OpenTEE	77
Figure 24 Consent Management Back-end architecture	78
Figure 25 Create new consent policy page.....	79
Figure 26 Selection of specific identity attribute.....	80
Figure 27. Selection of identity attributes below a certain LoA	80
Figure 28. Selection of specific Identity Provider	80
Figure 29. Selection of Identity Providers below a certain LoA.....	81
Figure 30. Selection of specific Service Provider	81
Figure 31. Selection of Service Providers below a certain LoA.....	81
Figure 32. List of a user's identity attributes	82
Figure 33. View created policies per identity attribute	82
Figure 34. List of Identity Providers	83
Figure 35. List of created policies per Identity Provider	83
Figure 36. List of Service Providers	84
Figure 37. List of created consent policies per Service Provider	85
Figure 38 Consent Management Web Front-end architecture	86
Figure 39 P-ABAC-FIDO Proposed Integrated Architecture	91
Figure 40 Authentication process from the FIDO UAF specification	92
Figure 41 P-ABAC-FIDO integrated authentication protocol - proposed changes to the FIDO UAF specification are highlighted in red	93
Figure 42: Credential Backup & Restore mobile application: Main page	95

Figure 43: List with the Cryptographic Credentials stored in the mobile device.....	96
Figure 44: Page that shows to the user the details of a cryptographic credential	97
Figure 45: Details of an encrypted cryptographic credential	97
Figure 46: List with the Cryptographic credentials that are backed up in the Identity Consolidator ..	98
Figure 47: Process of fetching a cryptographic credential from the Identity Consolidator server	98
Figure 48: View list of user's backed-up credentials.....	99
Figure 49: Search user's credentials	99
Figure 50: View the details of a credential	100
Figure 51: Delete a credential.....	100
Figure 52 Protocol to integrate OpenAM within P-ABAC architecture	102
Figure 53 Mapping of the IRMA Verification Protocol into FIWARE Open RESTful APIs	103
Figure 51 screen shot of the Create ABAC Policy Tab.....	105
Figure 52 Show ABAC Policies tab.....	105
Figure 53 AccBAC Policies creation	106
Figure 54 AccBAC Policies View	106
Figure 55 PDP Request Example	106
Figure 56 PDP Policy Example	107
Figure 57 Database table: requests	108
Figure 58 Database table: policies	108
Figure 62 CP-ABE based proposed ABAC mechanism	113
Figure 63 WI-FAB overview diagram.....	114
Figure 64 Format of the Vendor Specific Information Elements included in the IEEE 802.11 beacons broadcast by the access point.....	115
Figure 65 WI-FAB implementation overview.....	115
Figure 66 ECDF associated to the number of collected beacons needed to reconstruct the encrypted secret with and without fountain coding (FC)	117
Figure 67 Time needed for the station to connect vs. number of attributes in the AP policy	117
Figure 68 Time needed for the station to connect vs. random WPA2 key regeneration interval (500 connection attempts for each regeneration interval, policy with 4 attributes)	117

1 Introduction

The main target of the ReCRED project is to design and implement an architecture that allows a user to simplify online identities management by consolidating them and by exploiting the concept of *Device Centric Authentication (DCA)*. However, another main objective of the project is to go beyond the state-of-the-art of Device Centric Authentication platforms by integrating privacy-aware access control capabilities, which are lacking in actual DCA platforms. Indeed, this is a natural step since user identities are basically formed by user attributes that can be exploited in order to realize an Attribute Based Access Control (ABAC) on resources that the user wants to access. Moreover, since the ReCRED project targets mainly the security of the users' identity and privacy in general, it is needed to add privacy-preserving capabilities to the classical ABAC definition and technologies. This motivates the strong focus of the ReCRED project on the **integration** and **deployment** of state-of-the-art anonymous credentials platforms such as Idemix [43], U-Prove [45] and ABE [30] to realize a Privacy-Preserving Attribute Based Access-Control (P-ABAC) architecture that is able to guarantee the anonymity of the involved users. The final target is to have as outcome of the project a reference ABAC architecture ready for the market, research projects and industrial exploitation.

The integration of such anonymous credential systems (Idemix and U-Prove), enriched with the state-of-the-art of cryptographic protocols like Attribute Based Encryption (ABE) is realized by means of the FIWARE Privacy Open RESTful API [49]. This enables ReCRED to implement the ABAC architecture exposing to the developer a single interface to be exploited for all these protocols. Moreover, our ABAC infrastructure is designed to be completely integrated in commercial and widely used authentication systems like FIDO and OpenID connect (OpenAM implementation).

The ABAC architecture will maintain the main device centric approach of the project thanks to the integration of these ABAC systems in the user's device, supported by the deployment and execution inside the trusted execution environment (TEE) on the user-device. To reach such target, the ReCRED project is investigating existing and open source TEE platforms in order to provide a full and secure implementation of ABAC technologies directly on the device.

1.1 Attribute-Based Access Control

One of the purposes of the authentication process for users requesting to perform an action in a system (e.g. requesting or modifying a resource) is the effective verification of the permissions that such user has with respect to the specific requested action. Indeed, very often the concept of authentication is confused and mixed with the identification or the authorization. They are all distinct concepts, and should be thought of as such. Identification is nothing more than a user claiming it is somebody, like for example declaring a username. Authentication is how a user proves that she is who she declares to be: the user demonstrates the knowledge of something that only she knows (e.g. a password) or something that only she has (e.g. a private key). Authorization is what takes place after a user has been both identified and authenticated: it is this step that establishes what the user can do once identified and authenticated in the system. Access control is one of the functionalities that enables the verification of authorizations of a user when he/she is requesting to perform an operation over a resource.

Traditionally, access control is based on the identity of a user or pre-defined attribute types such as roles or groups assigned to that user. This approach requires cumbersome management, since it

requires to associate capabilities (authorized operations) directly to users or their roles or groups. Moreover, the requested qualifiers of identity, groups, and roles are often insufficient in the expression of real-world access control policies. An alternative is to grant or deny user requests based on arbitrary attributes of the user and arbitrary attributes of the resource, and environment conditions that may be globally recognized and more relevant to the policies at hand, that takes the name of *Attribute Based Access Control* (ABAC). ABAC enables resource and service providers to apply an access control policy without prior knowledge of the specific requester and for an unlimited number of users that might require access. As a new user joins the system, rules and resources do not need to be modified. Since the user is assigned the attributes necessary to access the required objects no modifications to existing rules or object attributes are required.

The evolution of access control models starts from the definition of *Mandatory Access Control* (**MAC**) and *Discretionary Access Control* (**DAC**) [64][65]. Such kind of access control was first implemented in the U.S.A. Department of Defense (DoD) applications and often employed in government and military facilities. Mandatory access control works by assigning a classification label (including *confidential*, *secret* and *top secret*) to each file system object. Each user and device on the system is assigned a similar classification and clearance level. While it is the most secure access control setting available, MAC requires careful planning and continuous monitoring to keep all resource objects' and users' classifications up to date.

As the highest level of access control, MAC can be contrasted with lower-level discretionary access control (DAC), which allows individual resource owners to make their own policies and assign security controls.

As networks and systems grew, the need to limit access to specific protected objects spurred the growth of *Identity-Based Access Control* (**IBAC**) that employs mechanisms such as access control lists (ACLs) to capture the identities of those allowed to access the object. If a subject presents a credential that matches the one held in the ACL, the subject is given access to the object. Individual privileges of the subject to perform operations (read, write, edit, delete, etc.) are managed on an individual basis by the object owner. Each object needs its own ACL and set of privileges assigned to each subject. In the IBAC model, the authorization decisions are made prior to any specific access request and result in the subject being added to the ACL. For each subject to be placed on an ACL, the object owner must evaluate identity, object, and context attributes against policy governing the object and decide whether to add the subject to the ACL. This decision is static and a notification process is required for the owner to re-evaluate and perhaps remove a subject from the ACL to represent subject, object, or contextual changes. Failure to remove or revoke access over time leads to users accumulating privileges.

The most important improvement on the management of permissions and access control policies is provided by the *Role-Based Access Control* model (**RBAC**) [66][67][68] that employs pre-defined roles carrying a specific set of privileges associated with them and to which subjects are assigned. For example, a subject assigned the role of Manager will have access to a different set of objects than someone assigned the role of Analyst. In this model, access is implicitly predetermined by the person assigning the roles to each individual and explicitly by the object owner when determining the privilege associated with each role. At the point of an access request, the access control mechanism evaluates the role assigned the subject requesting access and the set of operations this role is

authorized to perform on the object before rendering and enforcing an access decision. Note that a role may be viewed as a subject attribute that is evaluated by the access control mechanism and around which object access policy is generated. As the RBAC specification gained popularity, it made central management of enterprise access control capabilities possible and reduced the need for ACLs.

ACLs and RBAC are in some ways special cases of *Attribute-Based Access Control (ABAC)* in terms of the attributes used. ACLs work on the “identity” attribute while RBAC works on the “role” attribute. The key difference with ABAC is the concept of policies that express a complex *Boolean* rule set that can evaluate many different attributes. While it is possible to achieve ABAC objectives using ACLs or RBAC, demonstrating access control requirements compliance is difficult and costly due to the level of abstraction required between the AC requirements and the ACL or RBAC model. Another problem with ACL or RBAC models is that if the AC requirement is changed, it may be difficult to identify all the places where the ACL or RBAC implementation needs to be updated.

Trying to implement IBAC or RBAC access control decisions would require the creation of numerous roles that are ad-hoc and limited in membership, leading to what is often termed “role explosion”. A method is needed to make access control decisions without previous knowledge of the object by the subject or knowledge of the subject by the object-owner. By relying upon the concepts of subject and object attributes consistently defined between organizations, ABAC avoids the need for explicit authorizations to be directly assigned to individual subjects prior to a request to perform an operation on the object. Moreover, this model allows flexibility in a large enterprise where management of access control lists or roles and groups would be time consuming and complex. Leveraging consistently defined attributes that span both subjects and objects, authentication and authorization activities can be executed and administered in the same or separate infrastructures, while maintaining appropriate levels of security. In general, ABAC avoids the need for capabilities (operation/object pairs) to be directly assigned to subject requesters or to their roles or groups before the request is made. Instead, when a subject requests access, the ABAC engine can make an access control decision based on the assigned attributes of the requester, the assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions. Under these arrangement policies can be created and managed without direct reference to potentially numerous users and objects, and users and objects can be provisioned without reference to policy.

In addition to the high flexibility of the ABAC approach for access-control, we strongly believe that the users’ privacy and anonymity should be guaranteed by an ABAC architecture. Given this motivation, the ReCRED project designed the access-control around the concept of Privacy-Preserving Attribute Based Access-Control (**P-ABAC**) where, the user is authorized based only on effectively required attributes without requiring the disclosure of its identity. This is possible thanks to the integration of cryptographic schemes like Idemix [43], U-Prove [45] and Attribute Based Encryption (ABE) [30].

1.2 Integration of ABAC in ReCRED

The state-of-the-art of *Device Centric Authentication (DCA)* platforms does not provide such access control capabilities since they are only focused on providing authentication methods to the user. The ReCRED platform will fill this gap by integrating support for attribute-based access control (ABAC) allowing to support the validation of the complete identity of an individual when the verifier considers only a specific identity attribute in order to grant the individual access to a resource. The ReCRED architecture for ABAC mainly addresses the fragmentation of the access control models and aims to

create a platform that is open, i.e., designed and developed in such a way that it may interact with all the existing and new emerging standards and solutions in the area of user identification, authentication and access management.

A key goal of the entire ReCRED platform is to guarantee the protection of the privacy of the individuals using the services. In order to match this goal, the ReCRED ABAC architecture adopts a private credentials approach to let users prove their identity attributes safely from their device to the online or physical relying service. Indeed, in most of the real-world scenarios, users do not need to reveal their complete identity as the verifiers require knowing only an aspect of their identity (e.g., their age, their home address, their profession, whether they are students, etc.). Yet, users are forced to reveal their credit card details or present ID cards. By facilitating attribute-based access control, our solution becomes privacy-preserving by design thanks to the integration of Anonymous Credential Systems, i.e. cryptographic protocols that can support attribute-based access control (ABAC) like Idemix [43] and U-Prove [45].

In order to provide a fast implementation and deployment of the ABAC, ReCRED gave particular attention to the results of relevant EC-funded projects like ABC4Trust [50] and FIWARE [48]. The first provides a baseline architecture for Anonymous Credential System to be used in the P-ABAC architecture, while the second allows to use a common protocol between different Anonymous Credentials Systems.

One of the ambitious goals of ReCRED project, discussed in this document, is the integration of the ABAC framework with most common and widely deployed technologies for user authentication and authorization like FIDO [38] and OpenID connect [39]. The ReCRED P-ABAC infrastructure provides such integration thanks to specific protocol and components designed to adapt the credential system with common username/password or public key systems.

2 P-ABAC Architecture

The ReCRED Device-Centric Reference Architecture described in Deliverable D2.3 and partially depicted in Figure 2 includes sub-components in charge of the Attribute Based Access Control credentials management in order to provide functionalities for the issuing and the verification of such credentials as well as the secure storage and management.

The architectural design, as well as the protocols design, to support the ABAC infrastructure in the ReCRED framework are driven by the following considerations: i) the protection of the privacy and the anonymity of users should be guaranteed, ii) the attributes issued to the users should be verified against certified real identity-attributes where it is possible, iii) the verification of attributes should occur only on really required attributes and, iv) the user should be able to consent the disclosure of attributes required to access a service.

The privacy of the users is guaranteed by means of cryptographic approaches to realize the Privacy-Preserving Attribute-Based Access Control (P-ABAC). Idemix and U-Prove are the baseline of our P-ABAC architecture that allows to have a fully anonymous, un-linkable and un-trackable credential system. All components have a standard interface for the exchange of information provided by FIWARE Privacy Open RESTful API specifications [49]. This section will provide to the reader a detailed description of all components involved in the final design of the P-ABAC architecture supported in the ReCRED framework.

2.1 P-ABAC Architectural Overview and Relation to the ReCRED Architecture

2.1.1 ABAC Components

The ReCRED P-ABAC architecture does not differ from a standard ABAC architecture where three main actors are involved:

- **Issuer:** similar to the role of an Authority in standard PKI infrastructures, is the component in charge for releasing (issuing) credentials to users.
- **Recipient/Prover:** is the user that collects credentials and use them to generate the proof of possession to requesting services.
- **Verifier:** is the component that requires the proof of possession of attributes of the user. It usually requires from the user to prove the possession of attributes that belong to a Boolean policy.

A more detailed description of these components from a functional point of view is provided in Figure 1 and in the following sub-sections, while a description of the actual mapping of these components to the ReCRED general architecture is included in Section 2.1.2.

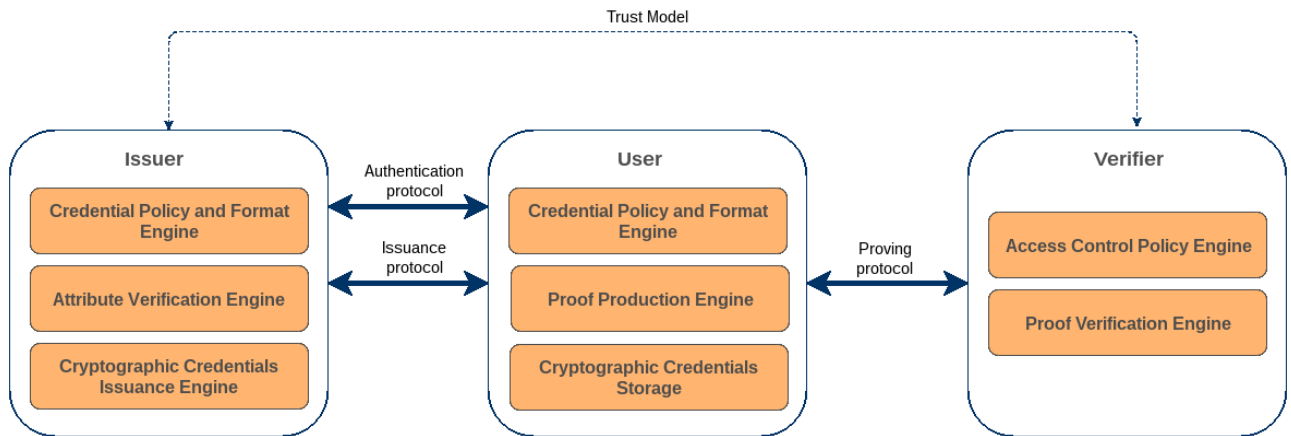


Figure 1 ReCRED ABAC functional architecture with elements involved

2.1.1.1 User

In the ReCRED P-ABAC architecture, the User is the subject that wants to access a resource. It owns verifiable attributes and can hold and receive credentials from the Issuer. It can securely authenticate to Issuers and correctly parse credential policies. It can also send to Verifiers certified proofs (which can be zero-knowledge proofs) of possession of its attributes.

These functionalities are supported by:

- the User Credential Policy and Format Engine, which can correctly interpret the semantics of the credential policies and specifications provided by the Issuer
- the Proof Production Engine, which can output partial verifiable profiles including only a subset of the User attributes or a set of assertions based on them
- the Cryptographic Credentials Storage, which can securely store the cryptographic credentials obtained by the issuer

The User takes part both in the Issuing and Authentication protocols together with the Issuer and in the Proving protocol with the Verifier.

2.1.1.2 Issuer

In the ReCRED ABAC architecture, the Issuer is the entity that is able to verify the attributes of the User and to issue credentials that certify these attributes.

These functionalities are supported by:

- the Issuer Credential Policy and Format Engine, which can associate policies and credential types with the issuance action initiated by the User
- the Attribute Verification Engine, which can verify the attributes owned by the credential requesting User
- the Cryptographic Credential Issuance Engine, which can produce cryptographic credentials based on the verified attributes of the User

The Issuer takes part in the Authentication and Issuing protocols together with the User.

2.1.1.3 Verifier

In the ReCRED ABAC architecture, the Verifier is the entity that holds a resource that the User wants to access. It can verify the proofs provided by the User according to given policies, provided that a chain of trust exists towards the credential issuer.

These functionalities are supported by:

- the Access Control Policy Engine, which provides the attribute-based policies associated to requested resources
- the Proof Verification Engine, which can verify, according to a given policy, the partial verifiable profiles provided by the User

The Verifier takes part in the Proving protocol together with the User.

2.1.2 ReCRED Components Mapping to the ABAC Architecture

This section describes how the ABAC components described above (User, Issuer and Verifier) are mapped to the ReCRED Reference Architecture described in Deliverable D2.3. To this aim, Figure 2 shows the ReCRED architecture from an ABAC perspective.

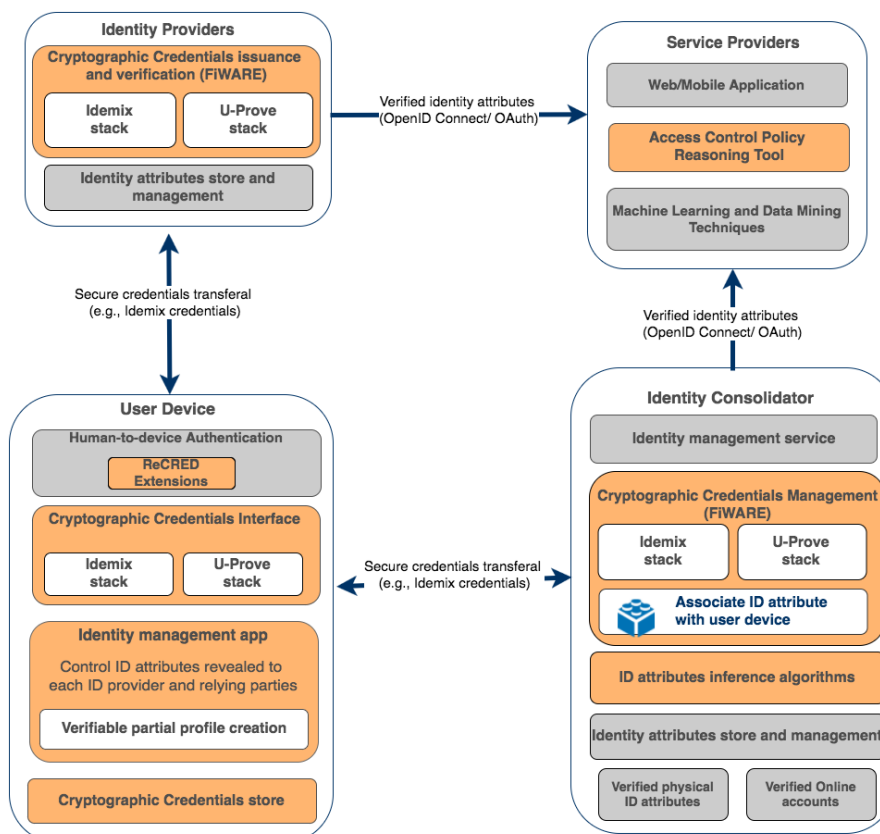


Figure 2 ABAC components view of the ReCRED Architecture

2.1.2.1 User Device

The User Device is the central component of the ReCRED architecture. From the ABAC point of view it maps to the User functionality described in Section 2.1.1.1. It has the capability to securely

authenticate to Issuers and the Identity Consolidator (e.g. through FIDO [38], OpenID [39], OAuth [40]), to request cryptographic credentials from Issuers and the Identity Consolidator, to securely store cryptographic credentials, to backup credentials in the Identity Consolidator, to correctly parse and show Issuer and Verifier policies to the user and to create verifiable partial profiles containing a subset of identity attributes.

The Cryptographic Credentials Storage functionality is provided in the ReCRED architecture by the component with the same name, while the Proof Production Engine and the User Credential Policy and format engine functionalities are provided by the Identity Management application.

2.1.2.2 Identity Consolidator

The Identity Consolidator is the ReCRED component that enables horizontal identity binding and vertical real-to-online identity mapping. It can acquire the physical attributes of real world identities. From the ABAC point of view it maps to the Issuer functionality described in Section 2.1.1.2. It has the capability to allow secure authentication from the User Device, to issue verified cryptographic credentials to the user device, to allow backup credential storage from the user device, to prove identity attributes on behalf of the user and to receive identity attributes from the Identity Providers.

The Attribute Verification Engine, Issuer Credential Policy and Format Engine and Cryptographic Credential Issuance Engine functionalities are provided by the Cryptographic Credentials Issuance and Revocation module.

2.1.2.3 Identity Providers

In the ReCRED architecture the Identity Providers are the entities in charge of managing the identity of the users. Such entities are able to verify the identity of the users in order to issue credentials accordingly. Moreover, they provide functionalities for PABAC credentials verification to the Service Providers (SP). In what follows the two functionalities provided by the IdP are discussed.

2.1.2.3.1 Issuing Authorities

From the ABAC point of view, the Issuing Authorities map to the Issuer functionality described in Section 2.1.1.2. They are able to allow secure authentication from the User Device, to issue credentials to the User Device, to transfer issued credentials as a backup to the Identity Consolidator on behalf of the user and, when privacy-aware cryptographic protocols are not supported, to transfer identity attributes to the Identity Consolidator so that this can issue credentials on behalf of the Identity Provider.

The Attribute Verification Engine, Issuer Credential Policy and Format Engine and Cryptographic Credential Issuance Engine functionalities are provided by the Cryptographic Credentials Issuance module.

2.1.2.3.2 Verifiers

From the ABAC point of view they map to the Verifier functionality described in Section 2.1.1.3. They have the capability to provide complex access-right policies to the users, to verify cryptographic credentials against these policies and to accordingly grant or deny access to the resources.

The Access Control Policy Engine and Proof Verification Engine functionalities are provided by the Access Control Policy and ReCRED Daemon modules.

2.2 Detailed P-ABAC Architectural Description

2.2.1 Privacy Preserving Attribute-Based Credential Systems

2.2.1.1 Idemix

Idemix [41][42][43] is an identity management system based on anonymous credentials and zero-knowledge protocols. Parties involved in the Idemix system can play the roles of issuers, recipients, provers and verifiers. The issuer represents the authority demanded to the issuance of credentials (for which she is responsible) to a recipient through an issuance protocol that results in the recipient owning a credential. When a proof of possession of credential is required by a verifier, the credential owner (the recipient that obtained the credential issued by the authority) acts in the role of a prover towards the verifier. An Idemix credential consists of a set of attribute values as well as cryptographic information that allows the owner of the credential to create the proof of possession. The components of the ReCRED reference architecture are easily mapped into the above described roles of the Idemix architecture, in order to realize an anonymous credential system to provide Attribute Based Access Control feature in the ReCRED framework. In this section, we are going to present the protocols and the messages involved in the Idemix system described in [41] to which we refer for a complete description of the cryptographic system.

2.2.1.1.1 Idemix Anonymous Credential Scheme

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

2.2.1.1.2 System Setup

The anonymous credential system requires general parameters, which are separated into system parameters consisting of bit lengths and group parameters which define the groups that are used within the underlying cryptographic scheme.

- **Global Parameters:** System parameters should be fixed and made public to all parties. Such global parameters include bit lengths and groups size to be used in the scheme. Such parameters’ values are reported in [41].
- **Issuer Parameters:** The issuer’s key pair is used for issuing credentials to the users, i.e. issuing signatures on a list of attributes requested by a user. The maximum number l of attributes of the credential is determined by the public key of the issuer. The number of attributes available to users is $l - \ell_{res}$ since some attributes, e.g., the master secret that is considered as an attribute, are reserved. The issuer generates:
 - A *safe RSA key-pair*: generates the safe primes p and q where $p = 2p' + 1$ and $q = 2q' + 1$, and then computes the RSA module $n = pq$
 - Random values $x_z, x_{R_1}, \dots, x_{R_l} \xleftarrow{R} \{2, p'q' - 1\}$ and $S \xleftarrow{R} QR_n$ in order to compute the CL signature [42] [44] parameters:

$$Z = S^{x_z} \text{ and } R_i = S^{x_{R_i}}$$

Finally, the issuer’s public key is the set of parameters $pk = (n, S, Z, R_1, \dots, R_l, P)$, where P is the proof of correctness of the public key, while the private key is $sk = (p, q)$.

- **User Parameters:** The user’s master secret m_1 is an integer chosen uniformly at random from the interval $[1, p]$.
- **Pseudonyms:** The user can generate as many pseudonyms (nym) and domain pseudonyms ($dnym$) as she wants. Each pseudonym or domain pseudonym is un-linkable to any other pseudonym or domain pseudonym generated by the user. However, the domain pseudonyms enforce that a user can only generate one pseudonym per domain (i.e., given the domain and the user’s master secret key, the domain pseudonym is unique).

2.2.1.1.3 Credential Issuance

The issuance phase is performed by running an interactive protocol between the Issuer and the Recipient (i.e. the User requesting the issuance of a credential owned and specified by the issuer she is contacting).

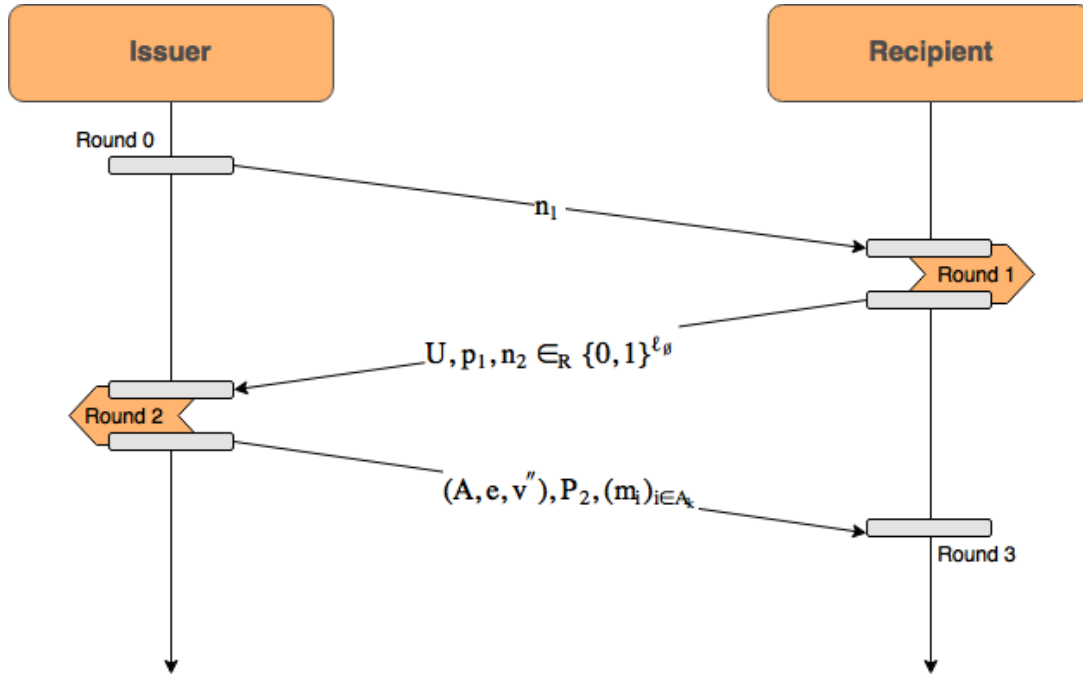


Figure 3 Idemix Issuance Protocol rounds between Issuer and Recipient components

2.2.1.1.3.1 Credential Issuance: Round 0

As shown in Figure 3, the issuance protocol is the result of four different phases, namely rounds, performed by the two parties that exchange the output of each round in order to proceed with the following one:

Round 0	Nonce Generation
Performed by	Issuer
Input	-
Output	n_1

2.2.1.1.3.2 Credential Issuance: Round 1

The *Issuer* extract a random value n_1 to be provided to the *Recipient* and start the issuance protocol by using such a fresh value.

$$n_1 \xleftarrow{R} \{0,1\}^{\ell_0}$$

Round 1	Attributes' commitment
Performed by	Recipient
Input	n_1
Output	U, P_1, n_2

The *Recipient* computes the value, considering the value of each attribute $\{m_i\}$:

$$U = S^{v'} \cdot \prod_{j \in A} R_j^{m_j} \bmod n, \text{ where: } v' \xleftarrow{R} \pm \{0,1\}^{\ell_n + \ell_0}$$

This is the commitment of the attribute's values m_i to demonstrate to the issuer the ownership of such attributes by the recipient. As output of this round, the *Recipient* produces a non-interactive proof P_1 of the above computation:

$$P_1 = \left(\begin{array}{l} c = H(\text{context} | C_1 | \dots | C_k | nym | dnym | \tilde{U} | \tilde{C}_1 | \dots | \tilde{C}_k | \widetilde{nym} | \widetilde{dnym} | n_1) \\ s_A = \{\hat{m}_0, \dots, \hat{m}_k\} \text{ where } \hat{m}_j = \tilde{m}_j + cm_j \\ \hat{v}' = \tilde{v}' + cv' \text{ where} \\ \hat{r}_j = \tilde{r}_j + cr_j \text{ if } nym \neq \perp \end{array} \right)$$

The values included in P_1 are reported below:

1. Choose the random value:

$$\tilde{m}_j \xleftarrow{R} \{0, 1\}^{\ell_0 + \ell_m + \ell_H + 1} \text{ for each attribute value}$$

2. Proof of the knowledge of pseudonym and master secret m_1 :

$$\widetilde{nym} = g^{\tilde{m}_1} h^{\tilde{r}_1} \text{ mod } \Gamma, \text{ where } \tilde{r}_1 \xleftarrow{R} [0, \rho]$$

3. Proof of the knowledge of domain's pseudonym and master secret m_1 :

$$\widetilde{dnym} = g_{dom}^{\tilde{m}_1} \text{ mod } \Gamma$$

4. Knowledge of representation of U :

$$\tilde{U} = S^{\tilde{v}'} \cdot \prod_{j \in A} R_j^{\tilde{m}_j} \text{ mod } n \text{ where } \tilde{v}' \xleftarrow{R} \pm \{0, 1\}^{\ell_n + 2\ell_0 + \ell_H}$$

5. Knowledge of committed values:

$$\tilde{C}_j = \{\tilde{C}_1, \dots, \tilde{C}_k\} \text{ with } \begin{cases} \tilde{C}_k = Z_k^{\tilde{m}_k} S_k^{\tilde{r}_k} \text{ mod } n \\ \tilde{r}_j \xleftarrow{R} \{0, 1\}^{2\ell_0 + \ell_n + \ell_H} \end{cases}$$

6. Fiat-Shamir challenge:

$$c = H(\text{context} | C_1 | \dots | C_k | nym | dnym | \tilde{U} | \tilde{C}_1 | \dots | \tilde{C}_k | \widetilde{nym} | \widetilde{dnym} | n_1)$$

7. Responses to the challenge:

$$\hat{v}' = \tilde{v}' + cv'$$

$$s_A = \{\hat{m}_0, \dots, \hat{m}_k\} \text{ with } \hat{m}_j = \tilde{m}_j + cm_j$$

In addition to the U and the proof of it P_1 , the *Recipient* extract a random value n_2 to be used as a client generated *nonce* for the issuance session:

$$n_2 \xleftarrow{R} \pm \{0, 1\}^{\ell_0}$$

2.2.1.1.3.3 Credential Issuance: Round 2

Round 2	Signature generation
Performed by	Issuer
Input	$U, p_1, n_2 \in_R \{0,1\}^{\ell_0}$
Output	$(A, e, v''), P_2, \{m_i\}_{i \in A_k}$

The *Issuer* verifies the correctness of the P_1 value sent by the *Recipient* by computing the following:

1. Knowledge of pseudonym and master secret m_1 :

$$\widehat{nym} = nym^{-c} g^{\widehat{m}_1} h^{\hat{r}} \mod \Gamma$$

2. Knowledge of domain pseudonym and master secret m_1 :

$$\widehat{dnym} = dnym^{-c} g_{dom}^{\widehat{m}_1} \mod \Gamma$$

3. Representation of U :

$$\widehat{U} = U^{-c} (S^{\hat{v}'}) \cdot \prod_{j \in A} R_j^{\widehat{m}_i} \mod n$$

4. Knowledge of committed values:

$$\tilde{C}_j = \{\tilde{C}_1, \dots, \tilde{C}_k\} \text{ with } \hat{C}_k = c_k^{-c} Z_k^{\widehat{m}_k} S_k^{\hat{r}_k} \mod n$$

5. Challenge verification to be compared with c sent by *Recipient*:

$$\hat{c} = H(\text{context} | C_1 | \dots | C_k | nym | dnym | \widehat{U} | \hat{C}_1 | \dots | \hat{C}_k | \widehat{nym} | \widehat{dnym} | n_1)$$

In case $c = \hat{c}$, the correctness is verified and the *Issuer* proceeds with the generation of the CL-signature σ_{CL} on attribute's values:

$$\sigma_{CL} = \begin{pmatrix} A = Q^{e^{-1} \mod p'q'} \\ e \xleftarrow{R} [2^{\ell_e-1}, 2^{\ell_e-1} + 2^{\ell'_e-1}] \\ v'' = 2^{\ell_v-1} + \tilde{v} \text{ with } \tilde{v} \xleftarrow{R} \{0,1\}^{\ell_v-1} \end{pmatrix}$$

1. The value Q is computed as follows:

$$Q = \frac{Z}{US^{v''} \prod_{i \in A_k} R_i^{\widehat{m}_i}} \mod n$$

To proof the correctness of the computation, the *Issuer* compute the value P_2 :

$$P_2 = \begin{pmatrix} c' = H(\text{context} | Q | A | \tilde{A} | n_2) \\ s_e = e - ce' \mod p'q' \end{pmatrix}$$

1. Such computation also includes the definition of:

$$\tilde{A} = Q^r \mod n \text{ with } r \xleftarrow{R} \mathbb{Z}_{p'q'}^*$$

This phase of the issuance protocol ends with the *Issuer* sending to the *Recipient* the following values:

$$\sigma_{CL} = (A, e, v''), P_2, \{m_i\}_{i \in A_k}$$

2.2.1.1.3.4 Credential Issuance: Round 3

Round 3	Signature storage
Performed by	Recipient
Input	$(A, e, v''), P_2, \{m_i\}_{i \in A_k}$
Output	(A, e, v)

The last step of the issuance protocol, executed by the *Recipient*, starts with the verification of the CL-signature on attribute’s values produced by the *Issuer*. Indeed, the *Recipient* first compute the value:

$$v = v'' + v'$$

Using that value checks whether:

$$Q = \frac{Z}{US^v \prod_{j \in A_k} R_i^{m_i}} \bmod n := \hat{Q} = A^e \bmod n$$

If $Q = \hat{Q}$ it proceeds with the verification of the values in P_2 :

1. Computes the value of \hat{A} :

$$\hat{A} = A^{c' + s_e e} S^{v' s_e} \bmod n$$

2. Checks whether:

$$\hat{c} = H(\text{context} | Q | A | \hat{A} | n_2) := c'$$

Finally stores the signature of all attributes in the credential $\{m_i\}_{i \in A}$: A, e, v

2.2.1.1.4 Credential Proving

The credential proving procedure, differently from the issuance protocol, is performed in only two steps, as shown in Figure 4. The credential proving phase aims to demonstrate to the verifier that a prover (i.e. the recipient of the issuance phase) effectively owns the combination of attributes that satisfy a given access control policy, namely *Statement S*, over such attributes. In the Idemix scheme, such policy is split in the so-called *Predicates* so that for each predicate the corresponding Prover and Verifier algorithm exists. For ease of discussion we will present here only details for the verification of a set of attribute’s values signature that is, de facto, the proof of possession of a set of attributes having specific values required by the verifier. The Idemix scheme allows also to prove and verify statement with more complex Boolean policies between attributes, as described in [42].

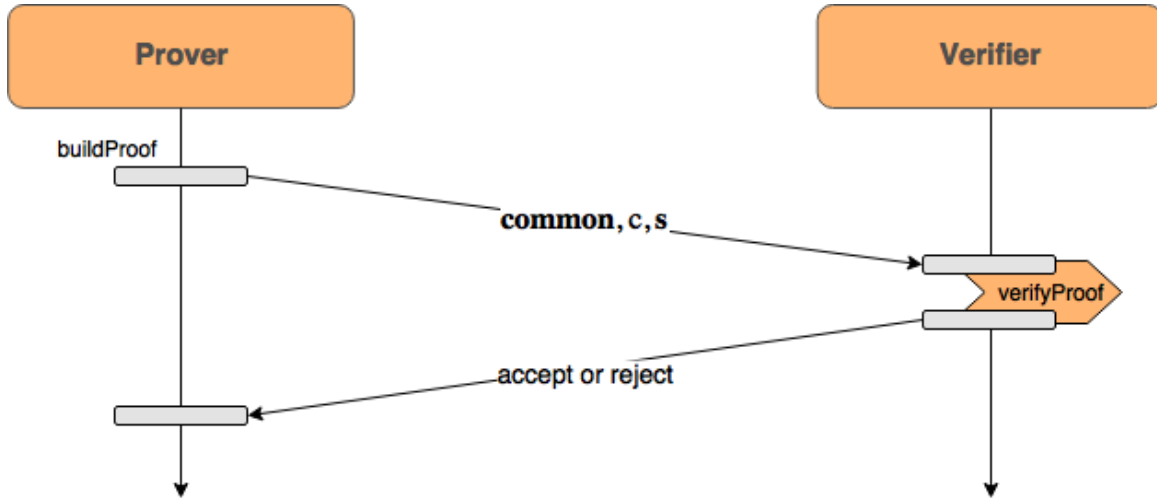


Figure 4 Idemix Proving Protocol rounds between Prover and Verifier components

This phase is split in a simple two-phase request-response protocol with the **buildProof** and the **verifyProof** procedures run, respectively, by the User/Prover and the Verifier.

2.2.1.1.4.1 Proof building procedure

buildProof	Produces the proof of possession of the credentials required by the verifier policy
Performed by	Prover
Input	$m_1, \{cred\}, S, n_1$
Output	non-interactive proof of statements in S : (Common, c, s)

Where S is the statement sent by the verifier, $\{cred\}$ are the credential(s) and the nonce n_1 is generated by the *Verifier* and sent to the *Prover*. In what follows we show the building procedure for the proof of the signature (sub-prover ProveCL) for a set of attributes A_r requested by and revealed to the *Verifier*. The attributes not requested by the *Verifier* will not be revealed to preserve the user’s privacy and reported as *hidden* attributes included in the set A_h . The prover maintains a global list of values that are common to all sub-provers algorithms of the build proof procedure:

1. Hidden value of each hidden attribute in A_h :

$$\tilde{m}_i \stackrel{R}{\leftarrow} \{0, 1\}^{\ell_0 + \ell_m + \ell_H}$$

2. Common values **common** and t-values **T** (ProveCL in the example):

1. Randomize the credential’s signature (A, e, v) :

$$r_A \stackrel{R}{\leftarrow} \{0, 1\}^{\ell_0 + \ell_n}$$

$$\sigma_R = \begin{pmatrix} A' = AS^{r_A} \\ v' = v - er_A \\ e' = e - 2^{\ell_e - 1} \end{pmatrix}$$

2. Compute t -values:

- $\tilde{e} \xleftarrow{R} \{0, 1\}^{\ell_0 + \ell_H + \ell'_e}$
- $\tilde{v}' \xleftarrow{R} \{0, 1\}^{\ell_0 + \ell_H + \ell_v}$
- For each attribute in A_h compute:

$$\tilde{Z} = (A')^{\tilde{e}} \left(\prod R_i^{\tilde{m}_i} \right) (S^{\tilde{v}'})$$

At this point add A' to the common values list **{common}** and \tilde{Z} to the t -values list $\{T\}$.

3. Compute the challenge:

$$c = H(\text{context} | \{\mathbf{common}\} | \{T\} | n_1)$$

4. Compute the s -values:

$$s = \begin{pmatrix} \hat{e} = \tilde{e} + c(e - 2^{\ell_e - 1}) \\ \hat{v} = \tilde{v} + cv' \\ \hat{m}_i = \tilde{m}_i + cm_i, \quad \forall m_i \in A_h \end{pmatrix}$$

The proof of possession of attributes A_r requested by the *Verifier* is: $(Common, c, s)$.

2.2.1.1.4.2 Proof verification procedure

verifyProof	is the verification phase of the protocol in which the verifier makes a decision on effective possession of required attributes by the prover based on the access policy (statement S) and the proof message. In the case of the proof of possession of specific attributes with specific values as stated above, the verifier will run the <i>VerifyCL</i> sub-prover
Performed by	Verifier
Input	$S, (Common, c, s), n_1$
Output	accept or reject of the proof

1. Retrieve the common values and the s -values and compute:

$$\hat{T} = \left(\frac{Z}{(\prod_{i \in A_r} R_i^{\hat{m}_i}) (A')^{2^{\ell_e - 1}}} \right)^{-c} (A')^{\hat{e}} \left(\prod_{i \in A_h} R_i^{\hat{m}_i} \right) (S^{\hat{v}'})$$

2. Add \hat{T} to the t -value verification list $\{\hat{T}\}$ and compute the challenge verification:

$$\hat{c} = H(\text{context} | \{\mathbf{common}\} | \{\hat{T}\} | n_1)$$

3. If the challenge verification \hat{c} matches the challenge c sent by the *Prover*, the verification is successful, otherwise reject it.

2.2.1.2 U-Prove

U-Prove [45] is a cryptographic protocol which assures the user's privacy by minimally disclosing the certified attributes while interacting with an on-line entity. U-Prove has three actors: the prover (user), the issuer and the verifier. The user to which is issued a cryptographic token interacts with the issuer and the verifier through an issuance and a presentation protocol. The U-Prove token is a container of attributes and it is digitally signed by the issuer entity. The token has a public key and a corresponding private key (both generated at issuance time) which must be kept secret by the prover. The private key can be used non-interactively by signing data which is later verified by the verifier or interactively in the presentation protocol where the prover signs a message in order to prevent replay attacks and demonstrate a proof-of-possession. The usage of the U-Prove token does not reveal the private key, thus mitigating attacks like eavesdropping or replay. The U-Prove technology provides both unlinkability and untraceability because the issuer executes a blind signature over the token and the prover demonstrates the possession of undisclosed attributes by executing a zero-knowledge protocol. Regarding the structure of the U-Prove token, it can contain a TI (Token Information) section and a PI (Prover Information) section. The TI section can be used as a metadata section making the token more informative: it can specify a validity period and is encoded by the Issuer, this section being always disclosed at presentation time. Being always disclosed in the presentation protocol, the PI section is encoded by the prover and is invisible for the issuer at issuance time. Each U-Prove token has a unique identifier, this information being used in identifying repeated visitors of an online service or for tracking revoked tokens. The U-Prove token identifier cannot be computed by the issuer, thus preserving the unlinkability and untraceability security attributes. The U-Prove technology permits the usage of a trusted device (smart card, mobile phone or even a trusted third-party server) on the prover side, when issuing a token. The device acts as a U-Prove token extension, having a private key and participating in the presentation protocol. The device can be used to extend multiple U-Prove tokens, even if issued by different issuers. Concerning the repeating visitor scenario, the prover can encode a pseudonym which permits the online service to recognize him even though using different U-Prove tokens.

2.2.1.2.1 U-Prove Primitives

2.2.1.2.1.1 Issuer Primitives

The issuer parameters have the following form:

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

UID_p is an octet string that holds an application-specific unique identifier for the Issuer parameters.

$desc(G_q)$ specifies a group G_q of prime order q .

UID_H is an identifier for the hash algorithm.

$(g_0, g_1, \dots, g_n, g_t)$ is the Issuer’s public key. To generate g_0 , the issuer uses a private key y_0 : $g_0 = g^{y_0}$

(e_1, \dots, e_n) is a list of byte values which state if the corresponding attribute values are hashed when computing the public key.

S is an application specific octet string for the issuer parameters.

The application specific value n indicates the number of attributes encoded in each token.

Both the Verifier and the Prover must evaluate the Issuer parameters in the following way:

Input

Group description: $desc(G_q) = (p, q, g)$

Public generators: $(g_0, g_1, \dots, g_n, g_t) \in G_q$

Verify

G_q (if it is a subgroup construction)

p and q are odd prime numbers

q divides p-1

$g \in G_q$ and $g \neq 1$

Verify public key elements

for $i \in 0, 1, \dots, n$, verify that $g_i \in G_q$ and $g_i \neq 1$

2.2.1.2.1.2 Device Parameters

Credential tokens can be device-protected and when so, the following is the required additional data:

$$g_d, x_d, h_d$$

$g_d \in G_q$ is the device generator and must be a generator of G_q

$x_d \in \mathbb{Z}_q^*$ is the device’s private key.

$h_d = g_d^{x_d} \in G_q$ is the device’s public key. This public key is known by the Prover and by the Issuer during the issuance protocol. Both the Prover and the Issuer must verify that the device public key is a valid element of G_q .

2.2.1.2.1.3 Token

The token has the following form:

$$UID_p, h, TI, PI, \sigma'_z, \sigma'_c, \sigma'_r, d$$

UID_p is an identifier for the issue parameters

$h \in G_q$ is the token public key. It must be an element of G_q

TI denotes the token information field. This section is always disclosed to the Verifier and contains information like token usage restrictions or metadata.

PI is the value of the prover information field. This section contains information asserted by the issuer and is hidden from the Issuer. PI is always revealed during token presentation.

$\sigma'_z \in G_q$ And $(\sigma'_c, \sigma'_r) \in Z_q$ form the issuer signature

Boolean d is used to indicate if token is protected by a device.

2.2.1.2.1.4 Token Private Key

The private key of the token is $\alpha^{-1} \in Z_q^*$, α being a secret generated by the Prover in the issuance protocol.

2.2.1.2.1.5 Token Public Key

The token public key has the following form:

$$h = (g_0 g_1^{x_1} \dots g_n^{x_n} g_t^{x_t} [g_d^{x_d}])^\alpha$$

$(g_0, g_1, \dots, g_n, g_t) \in G_q$ is the Issuer's public key.

α is a secret value generated by the prover.

$x_t \in Z_q$ is computed by hashing the issuance protocol version 0x01, a digest of the issuer parameters and the TI field.

g_d, x_d are present if the token is protected by a device.

$x_i \in Z_q$ is obtained from the corresponding attribute value A_i either by hashing it (if e_i is 0x01) or by encoding it directly (if e_i is 0x00).

The value of $x_t \in Z_q$ is computed in the following way:

Input

Issuer parameter fields: $UID_p, desc(G_q), UID_H, (g_0, g_1, \dots, g_n, g_t), (e_1, \dots, e_n), S$

Token information field: TI

Device protected boolean: d

[Device generator: g_d]

Computation

$$P = H(\text{UID}_p, \text{desc}(G_q), \text{UID}_H, \langle g_0, g_1, \dots, g_n, g_t, [g_d] \rangle, \langle e_1, \dots, e_n \rangle, S)$$

$$x_t = H(0x01, P, TI) \rightarrow Z_q$$

The values x_i are computed in the following manner:

Input

Issuer parameter fields: q, UID_H, e_i

Attribute value: A_i

Computation

If $e_i = 0x01$

If $A_i = \emptyset$ then $x_i = 0$

Else $x_i = H(A_i) \rightarrow Z_q$

Else if $e_i = 0x00$

Verify that $0 \leq A_i < q$

$x_i = A_i$

Else return error

Return x_i

2.2.1.2.1.6 Issuer Signature

The issuer signature is never seen by the Issuer and thus it cannot be used to link a specific token with an issuance protocol. The signature has the following parameters $\sigma'_z, \sigma'_c, \sigma'_r$ and is verified in the following way:

Input

Issuer parameter fields: $\text{desc}(G_q), \text{UID}_H, g_0$

Token fields: $h, \text{PI}, \sigma'_z, \sigma'_c, \sigma'_r$

Verification

Verify that $h \neq 1$

$$\sigma'_c$$

Verify that $\sigma'_c = H(h, PI, \sigma'_z, g^{\sigma'_r} g_0^{-\sigma'_c}, h^{\sigma'_r} (\sigma'_z)^{-\sigma'_c})$

2.2.1.2.1.7 Token Identifier

The token identifier is computed by hashing the token's public key and the Issuer's signature in the following manner:

Input

Issuer parameter field: UID_H

Token fields: $h, \sigma'_z, \sigma'_c, \sigma'_r$

Computation

$$UID_T = H(h, \sigma'_z, \sigma'_c, \sigma'_r)$$

2.2.1.2.2 U-Prove Protocols

2.2.1.2.2.1 Issuance Protocol

The issuance protocol requires a precomputation on both sides followed by the exchange of three messages.

The following parameters are common for both the Prover and the Issuer:

- Issuer parameters: $UID_p, desc(G_q, UID_H, (g_0, g_1, \dots, g_n, g_t), (e_1, \dots, e_n), S)$
- Attributes: $(A_1, \dots, A_n), TI$
- Device protected boolean: d
- [Device parameters: g_d, h_d]

2.2.1.2.2.2 Prover Precomputation

Input:

$$x_t := \text{ComputeXt}(IP, TI, d, [g_d])$$

$$x_i := \text{ComputeXi}(IP, A_i)$$

$$\gamma = g_0 g_1^{x_1} \dots g_n^{x_n} g_t^{x_t} [h_d]$$

Prover information field PI

Precomputation:

Generate α at random from Z_q^*

Generate β_1, β_2 at random from Z_q

$$h = \gamma^\alpha$$

$$t_1 = g_0^{\beta_1} g^{\beta_2}$$

$$t_2 = h^{\beta_2}$$

Compute $\alpha^{-1} \bmod q$

2.2.1.2.2.3 Issuer Precomputation

Input:

$$x_t = \text{ComputeXt}(IP, TI, d, [g_d])$$

$$x_i = \text{ComputeXi}(IP, A_i)$$

$$\gamma = g_0 g_1^{x_1} \dots g_n^{x_n} g_t^{x_t} [h_d]$$

Private key: $y_0 \in Z_q$

$$\sigma_z = \gamma^{y_0}$$

Precomputation:

Generate w at random from Z_q

$$\sigma_a = g^w$$

$$\sigma_b = \gamma^w$$

2.2.1.2.2.4 Exchanged Messages

A) The first message is sent from the Issuer to the Prover: $(\sigma_z, \sigma_a, \sigma_b)$

B) The second message is sent from the Prover to the Issuer: σ_c

$$\sigma'_z = \sigma_z^\alpha$$

$$\sigma'_a = t_1 \sigma_a$$

$$\sigma'_b = (\sigma'_z)^{\beta_1} t_2 \sigma_b^\alpha$$

$$\sigma'_c = H(h, PI, \sigma'_z, \sigma'_a, \sigma'_b) \rightarrow Z_q$$

$$\sigma_c = \sigma'_c + \beta_1 \bmod q$$

C) The third message is sent from the Issuer to the prover: σ_r

$$\sigma_r = \sigma_c y_0 + w \bmod q$$

D) The Prover generates the token

$$\sigma'_r = \sigma_r + \beta_2 \bmod q$$

$$\text{Verify that } \sigma'_a \sigma'_b = (gh)^{\sigma'_r} (g_0 \sigma'_z)^{-\sigma'_c}$$

$$\text{Token } T = \text{UID}_p, h, TI, PI, \sigma'_z, \sigma'_c, \sigma'_r, d$$

$$\text{Private key } \alpha^{-1}$$

In Figure 5 the issuance protocol steps are depicted.

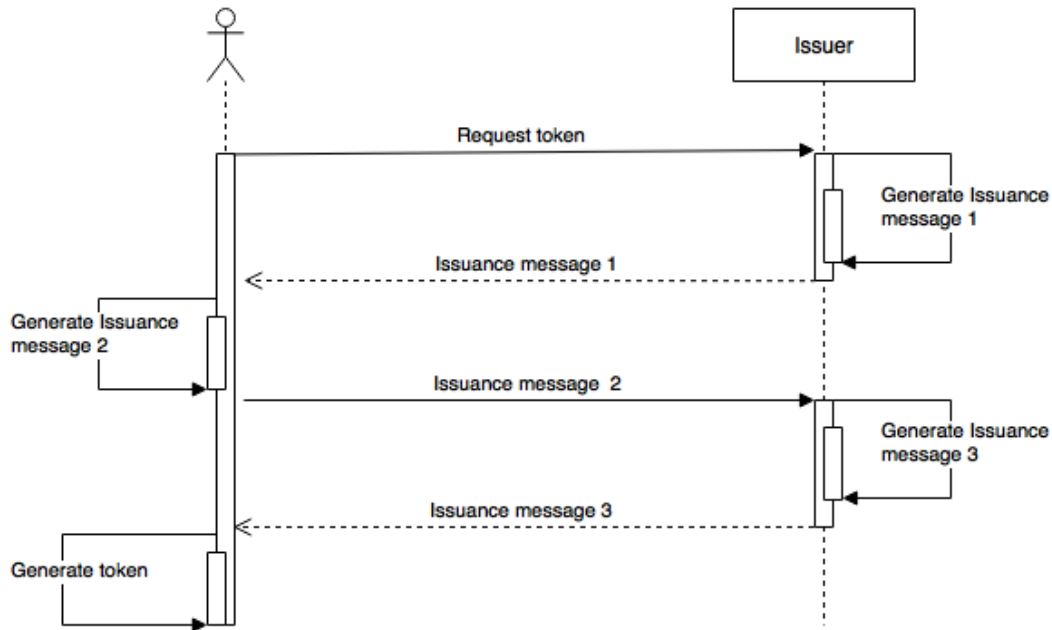


Figure 5 U-Prove Issuance protocol

2.2.1.2.2.5 Presentation Protocol

In the presentation protocol the Prover sends the token T , the subset of the attributes values it wants to disclose to the Verifier and a presentation proof generated by applying the token private key to a message and the non-disclosed attributes. The presentation proof is used to prove the integrity of the disclosed attributes and to prevent replay attacks.

The presentation protocol can be split in two steps: proof generation and proof verification.

2.2.1.2.2.6 Proof Generation

This step is a sub-protocol between the device and the Prover.

Device input:

Issuer parameters: $desc(G_q), UID_H$

Device generator: g_d

Private key: x_d

Prover input:

Issuer parameters: $UID_p, desc(desc(G_q)), UID_H, (g_0, g_1, \dots, g_n, g_t), (e_1, \dots, e_n), S$

Ordered indices of disclosed attributes: $D \subset \{1, \dots, n\}$

Ordered indices of undisclosed attributes: $U = \{1, \dots, n\} - D$

Ordered indices of committed attributes: $C \subset U$

Pseudonym attribute index: $p \in U \cup \{d\}$

Pseudonym scope: s

Messages: m, m_d

Token $T = UID_p, h, TI, PI, \sigma'_z, \sigma'_c, \sigma'_r, d$

Private key: α^{-1}

Attribute values: A_1, \dots, A_n

[Device generator: g_d]

Compute $x_i = \text{ComputeXi}(IP, A_i)$

Generate random $w_0, [w_d] \in Z_q$

For each $i \in U$ generate random $w_i \in Z_q$

A) The first message is sent from the Prover to the device: s

B) The second message is sent from the device to the Prover: $a_d, [a'_p, P_s]_{p=d}$

Generate w'_d at random from Z_q

$$a_d = g_d^{w'_d}$$

If $s \neq \emptyset$

$g_s = \text{GenerateScopeElement}(desc(G_q), s)$

$$a'_p = g_s^{w'_d}$$

$$P_s = g_s^{x_d}$$

C) The third message is sent from the Prover to the device: c_p, m_d

$$a := H\left(h^{w_0}(\prod_{i \in U} g_i^{w_i})[g_d^{w_d} a_d]_d\right)$$

If $p \neq \emptyset$ and $s \neq \emptyset$

$$g_s = \text{GenerateScopeElement}(\text{desc}(G_q), s)$$

$$a_p = H\left(g_s^{w_p}[a'_p]_{p=d}\right)$$

$$[P_s = g_s^{x_p}]_{p \neq d}$$

Else $a_p = \emptyset$ and $P_s = \emptyset$

For each $i \in C$

Generate σ_i, w_i at random from Z_q

$$c_i = g^{x_i} g_1^{\sigma_i}$$

$$\alpha_i = H(g^{w_i} g_1^{w_i})$$

$$UID_T = \text{ComputeTokenID}(IP, T)$$

If $p = d$ then $p' = 0$ else $p' = p$

$$c_p = H(UID_T, a, \langle D \rangle, \{\{x_i\}_{i \in D}\}, \langle C \rangle, \{\{c_i\}_{i \in C}\}, \{\{\alpha_i\}_{i \in C}\}, p', a_p, P_s, m)$$

$$c = H(\langle c_p, m_d \rangle) \rightarrow Z_q$$

$$r_0 = c\alpha^{-1} + w_0 \text{ mod } q$$

For each $i \in U$, $r_i = -cx_i + w_i \text{ mod } q$

D) The fourth message is sent from the device to the Prover: r'_d

$$c = H(\langle c_p, m_d \rangle) \rightarrow Z_q$$

$$r'_d = -cx_d + w'_d \text{ mod } q$$

E) The Prover created the presentation proof:

$$[r_d = r'_d + w_d \text{ mod } q]_d$$

For each $i \in C$, $\kappa_i = -c\sigma_i + w_i \text{ mod } q$

Presentation proof: $\{A_i\}_{i \in D}, a, (a_p, P_s), r_0, \{r_i\}_{i \in U}, [r_d]_d, \{(\sigma_i, \alpha_i, \kappa_i)\}_{i \in C}$

Secret commitment values: σ_i

2.2.1.2.2.7 Proof verification

Input:

Issuer parameters: $UID_p, desc(G_q), UID_H, (g_0, g_1, \dots, g_n, g_t), (e_1, \dots, e_n), S$

Ordered indices of disclosed attributes: $D \subset \{1, \dots, n\}$

Ordered indices of undisclosed attributes: $U = \{1, \dots, n\} - D$

Ordered indices of committed attributes: $C \subset U$

U-Prove token

Pseudonym attribute index: $p \in U \cup \{d\}$

Pseudonym scope: s

Messages: m, m_d

Presentation proof: $\{A_i\}_{i \in D}, a, (a_p, P_s), r_0, \{r_i\}_{i \in U}, [r_d], \{\epsilon_i, \alpha_i, \kappa_i\}_{i \in C}$

[Device generator: g_d]

Token verification:

$VerifyTokenSignature(IP, T)$

Presentation proof verification:

$x_t := ComputeXt(IP, TI, d, [g_d])$

For each $i \in D, x_i = ComputeXi(IP, A_i)$

$UID_T = ComputeTokenID(IP, T)$

If $p = d$ then $p' := 0$ else $p' := p$

$c_p := H(UID_T, a, \langle D \rangle, \langle \{x_i\}_{i \in D} \rangle, \langle C \rangle, \langle \{\epsilon_i\}_{i \in C} \rangle, \langle \{\alpha_i\}_{i \in C} \rangle, p', a_p, P_s, m)$

$c := H(\langle c_p, m_d \rangle) \rightarrow Z_q$

Verify that $a = H\left((g_0 g_t^{x_t} \prod_{i \in D} g_i^{x_i})^{-c} h^{r_0} (\prod_{i \in U} g_i^{r_i}) [g_d^{r_d}]\right)$

If $a_p \neq \emptyset$ and $P_s \neq \emptyset$

$g_s = GenerateScopeElement(desc(G_q), UID_H, s)$

Verify that $a_p = H(P_s^c, g_s^{r_p})$

For each $i \in C$, verify that $\alpha_i = H(\epsilon_i^c g^{r_i} g_1^{r_i})$

In Figure 6 are depicted the presentation protocols main steps.

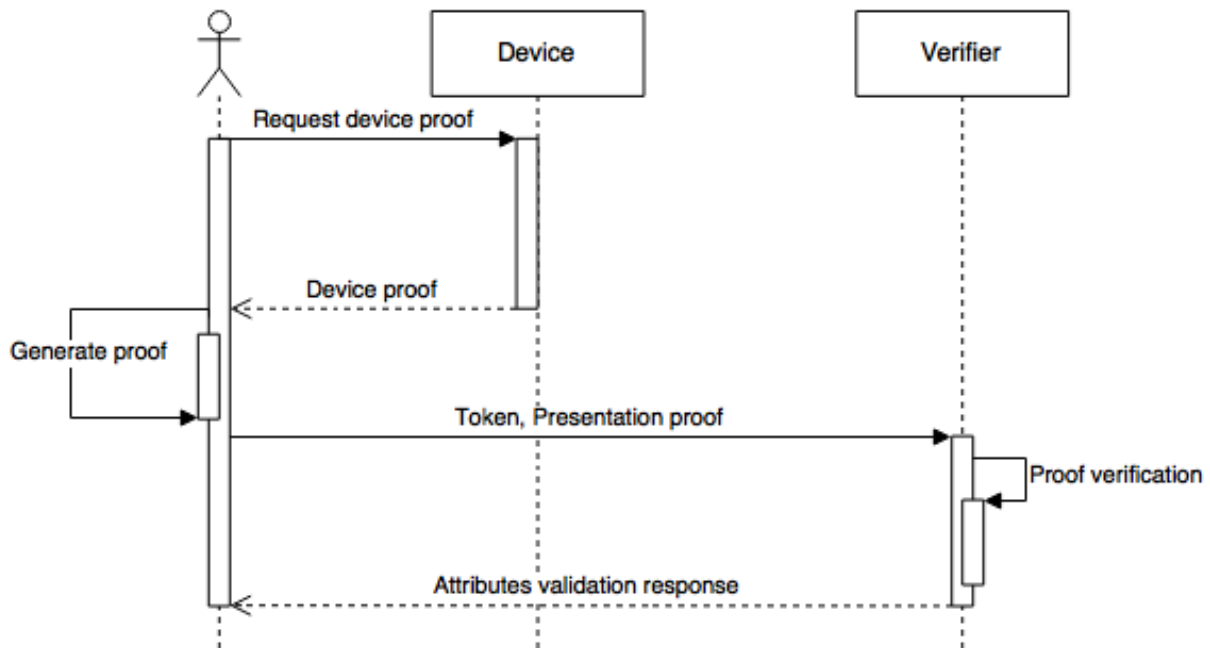


Figure 6 U-Prove Presentation Protocol

2.2.1.3 Attribute Based Encryption

Attribute Based Encryption (ABE) is an emergent new kind of asymmetric cipher. Similar to ordinary public key encryption schemes, a content is encrypted using a public key which does not reveal any information useful to decrypt the data, e.g. the private key. However, unlike ordinary public key schemes, the decryption key is not unique, but multiple users, having different keys, may decrypt the same message. Furthermore, and distinguishing feature of ABE, a user can decrypt a message only if the user is provided with a set of attributes which satisfy a given policy.

We are specifically interested in a variant of ABE called Ciphertext Policy, CP-ABE, where the policy which needs to be satisfied by the user’s attributes is directly integrated in the encrypted data itself, hence it “travels” with the data. Note that the combination of encryption (for confidentiality) and policy (for access control) in CP-ABE appears to be an extremely convenient approach for services that requires to release a specific resource outside of the trusted limits. This is best understood going into an example scenario using CP-ABE as cryptographic technique.

Let us assume that a content provider P wishes to encrypt a message M for a given set of users without the need to know a priori the identity of each individual user which shall be able to access the data, but wants to permit access to the data only to users which satisfy a given policy Π expressed in terms of attributes associated to the end users. Such a policy can be any arbitrary combination of “AND” and “OR” conditions, for instance

$$\Pi = (\text{italy:citizen AND job:executive}) \text{ OR } (\text{job:doctor})$$

Notably, at encryption time, the content provider only requires to know:

- the subset of attributes of interest, which are ordinary natural language strings
- the public key of the authority which has issued such attributes (as discussed later, CP-ABE was recently extended to operate with multiple non-coordinating authorities).

Not only CP-ABE does *not* require the content provider to a priori know the set of users which will be able to access the message, but it completely decouples the encryption process from the management of the user attributes. Indeed, suppose that CP-ABE encryption of a message M using policy Π occurs at a given time, say t_1 . Let $E[\Pi, M]$ be the resulting ciphertext, where we use a notation which highlights the fact that the policy Π is indeed integrated in the encrypted data itself. Suppose now that, at a subsequent time $t_2 > t_1$, a new user, say U_x , needs to be added to the set of users allowed to access the message. The user just need to retrieve the attributes required to decrypt and, such operation can be performed offline and once-for-all by contacting the related issuing authorities. The data itself does not require modification and continue to travel in the network or on untrusted storage without losing security capabilities.

2.2.1.3.1 A multi-authority CP-ABE architecture

The concept of CP-ABE has been originally introduced by Bethencourt, Sahai and Waters in 2007 [23]. This first construction however had a significant practical limitation in the fact that attributes were issued by a single, global, authority. In order to overcome such a limitation, the cryptographic community attempted to devise multi-authority CP-ABE schemes, with the first proposal in this field being a paper by Chase [46]. However, this first multi-authority proposal, as well as the subsequent extensions, still required some form of cooperation (at least offline) among the authorities. In the real world, such form of cooperation is deemed to be unviable, as it would force all possible authorities (ranging from banks, governments, visa offices, and even individuals) to interact at least once each other, as well as re-run a cooperation protocol every time a new authority is deployed. Also, mostly for this reason, CP-ABE did not have any notable practical success outside the restricted community of cryptographic researchers.

In a breakthrough paper, dated 2011 [47], Lewko and Waters proposed the first fully decentralized CP-ABE construction, thus broadly extending CP-ABE’s application range and make it fitting the real world needs of large scale networks and deployments. In this context, fully decentralized means that access policies can be specified over an arbitrary set of attributes issued by multiple independent and not cooperating authorities (possibly not even knowing each other’s existence). The far from being trivial technical challenge solved in [47] was the construction of a scheme resistant to collusion among users; in other words, if user U_1 holds attribute attr_1 issued by an authority A_1 and user U_2 holds attribute attr_2 issued by a *different* authority A_2 which has never cooperated or exchanged any information with A_1 , and even if the two users collude by exchanging their secrets associated to such attributes, as well as any other possible information locally held by the two users, it should be impossible (computationally hard) for each of these users to decrypt a message encrypted with the policy $\Pi = (\text{attr}_1 \text{ AND } \text{attr}_2)$. We refer the reader to the original work [47] for the cryptographic construction details. Despite the original construction [47] still suffers of some minor technical limitations, we believe that the notion of independent and fully decentralized authority therein exploited very well fits with the real-world needs.

Motivated by the availability of an actual, fully decentralized, multi-authority CP-ABE cryptographic construction, in what follows we preliminary sketch a multi-authority CP-ABE-based security architecture.

Attribute-issuing authorities. An authority A_i is any *arbitrary* entity (hence even including individual users) which autonomously decides to issue attributes. The set-up of an authority is thus an

independent decision, and does not require any coordination or interaction with a global authority. The only requirement an authority must adhere is to use a same set of globally-defined and publically known system parameters (in essence, a small set of standardized parameters, which, to make an illustrative example for the the specific CP-ABE setting of [47], appendix D, simply consist in a bilinear group \mathbb{G} of prime order p , in a generator g of the prime order group, and in a hash function H mapping global identity names into points of the group \mathbb{G}). An authority x will be characterized by a pair of keys: a public key $A_{x,PK}$, and an associated secret key $A_{x,SK}$. An authority is univocally identified by its public key: since this public key cannot be decided by the authority, but is computed by a cryptographic algorithm, the possibility that two authorities shall have the same PK is negligible. Although not technically necessary, if human readable names shall be used for authorities, an ordinary PKI must be supplementary used to bind the authority’s public key to its real world name, and avoid authority impersonation attacks. More formally, we summarize the set up of an authority with a publically known algorithm:

$$A_x.AUTHORITY_SETUP(\text{global parameters}) \rightarrow A_{x,PK}, A_{x,SK}$$

which is independently run by each authority, and which computes the authority’s public and private key pair.

Attributes. An attribute is a plain text string defined by, and associated to, an authority. For instance, an attribute can be as general as the string “visa” associated to a country-wide immigration authority and used to grant access permissions to a given country, or as specific as the string “office-mate” issued by an individual. Attributes shall not need to be globally unique (thus simplifying naming issues), but just need to be unique inside a same authority. For example, two countries (say Russia and Japan) can issue the same attribute string named “visa”, but these two attributes are different as they are issued by different authorities. Whenever ambiguity occurs, we will use the scope symbol “:” to differentiate the two attributes, e.g. Russia:visa versus Japan:visa, but we stress that this is just a notational convenience and not the bit string associated to the actual attribute (which, in both cases, it is simply the string “visa”).

Attribute-issuing procedure and Global identity names. In order to get an attribute from an authority, a user must have a global identity name, called UID (user ID), which must be a globally unique bit-string, for instance, an email address. Attributes issued to different identity names (even if belonging to a same human user, e.g. two different email addresses) will not be combined in a same access control policy. For instance, if the same human user holds two identity names, e.g. foo@mail.com holding attribute x and foo@recred.com being issued attribute y , the user will *not* be able to access a data encrypted with CP-ABE using the policy $(x \text{ AND } y)$. In order to be granted an attribute, a user will offline submit to an authority its global identity name, and if the authority decides to issue the required attribute, the user will receive back a secret key uniquely associated to both the user as well as the attribute. Note that this implies that different users will get different secrets for the same attribute. Formally, we summarize the attribute issuing procedure as an algorithm

$$A_x.ATTRIBUTE_ISSUING(\text{UID}, \text{attr}_j, A_{x,SK}) \rightarrow K_{\text{UID},\text{attr}_j}$$

Where UID is the global identity name of the user, attr_j is the issued attribute name, $A_{x,SK}$ is the Authority secret key, and $K_{\text{UID},\text{attr}_j}$ is the secret key released to the user for the considered attribute.

This algorithm shall be executed by the authority, and the resulting secret key shall be provided to the user via a secure channel.

Note that the compelling aspect of the above sketched architecture resides in the fact that it does not specify any necessary system component (e.g. unlike IPsec, where security associations require to be supported by security association databases and security policy databases). The trust model underlying the access control operation is mandated to individual trust relations (which can eventually, but not necessarily, exploit a certification PKI infrastructure) among entities and attribute-issuing authorities, rather than to a trust infrastructure. This can be very clearly highlighted through the following encryption use-case example. Assume that user U_x decides to share a message M encrypted with the policy

$$\Pi = (\text{Italy:citizen AND age:greater_than_18 AND } U_x\text{:friend}) \text{ OR } (\text{italian_police:officer})$$

where attributes are written using scope notation (i.e., authority:attribute). In order to encrypt message M , the user needs to decide/have:

- the attribute bit strings, i.e. “citizen”, “greater_than_18”, “friend”, and “officer”;
- the access control policy;
- the public keys of the four involved authorities, i.e.
 - a national authority from Italy which releases citizenship permissions;
 - an authority which certifies, by issuing a relevant attribute, that an user has an age greater than 18;
 - a national police authority, and
 - the user herself; indeed, since any entity can become authority, the user can as well decide to issue her own attributes, such as the “friend” attribute highlighted in the policy.

Once the message is encrypted, the user knows that the message will be accessed only by other users which have been issued a set of attributes by the specific authorities considered. Indeed, the encryption of a message is performed by ciphering the message using the attribute bit strings as well as the public keys of the relevant authorities which are in charge of issuing the given attributes. Note that this is a significant generalization of the ordinary asymmetric public key encryption, with the notable difference that the public key used during encryption is not anymore, the one of the recipient of the message, but are those of the attribute issuing authorities. In essence, in terms of trust, CP-ABE implies that the user just relies on her individual trust in the *specific* authorities involved, which are identified through their public keys.

Since Attribute Based Encryption schemes realize an implicit access control mechanism on the encrypted data, we believe that the ReCRED ABAC architecture could benefit of the usage of such techniques. Indeed, it can be used both to realize an access control on static data distributed in the network (data encryption) both an access control for the user (token encryption).

2.2.1.4 Credential Revocation

When using anonymous credentials in the real-world environment, there are additional requirements that should be considered in the design of the architecture. One of these requirements is the need to have the functionalities and entities to enable the revocation of the credentials issued to a user.

In traditional systems based on a Public Key Infrastructure (PKI) the common adopted approach is to publish a list (Certificate Revocation List - CRL) of serial numbers related to revoked certificates/credentials. Another approach, alternative to the CRL one, is to enable the authority/issuer to be queried by users about the effective validity of a specific certificate by running a protocol like the Online Certificate Status Protocol (OCSP). Unfortunately, this kind of approaches cannot work in systems like ReCRED and related Anonymous Credential Systems exploited to design the P-ABAC architecture. Indeed, it would compromise the anonymity and the privacy of the users in contrast with the ReCRED platform where the highest priority is to maintain the anonymity of the users.

In anonymous credential systems, indeed, the credential specific identifier is no longer revealed since this is one of the key requirements of anonymous credentials. In what follows we present some of the several revocation strategies [1] for anonymous credentials proposed in literature.

2.2.1.4.1 Verifiable Encryption

Although verifiable encryption is often cited in anonymous credential schemes related to anonymity revocation [2][3], it could be exploited to enable revocation capabilities. Hence, the user encrypts by using verifiable encryption the credential's identifier with the public key of the credential's issuer. To verify the revocation status, the verifier sends the ciphertext to the issuer. The issuer decrypts the ciphertext and is able to use the obtained identifier to do a simple lookup of the revocation status of the corresponding credential and provide the result back to the verifier. This solution is closely related to the OCSP protocol in traditional PKI systems. However, the issuer must act as a completely trusted party by the user's point of view, since it is able to monitor and track the usage of the credential (i.e. to which verifier the credential is shown). A possible solution is to require the service provider to make this request over an anonymous channel. Furthermore, replacing the public key of the issuer with the public key of another trusted third party, allows to have a separate authority in charge of the revocation tasks. Moreover, if the encrypted identifier is replaced with a domain specific pseudonym, a domain specific revocation authority may take care of access control in a certain domain. Another obvious drawback of this solution is given by the requirement of having all partners online when verifying credentials.

2.2.1.4.2 Limited Lifetime

In this approach, an attribute expressing the lifetime of the credential, is enclosed in the credential. When performing a proof of possession of the credential, the user also proves that the credential has not expired. The lifetime of a credential highly determines the usability of the revocation scheme:

- **Short lifetime** requires the user to frequently re-validate the credential and makes the scheme suitable to fit revocation requirements.
- **Long lifetime** makes the scheme insecure and not usable for revocation purposes.

Instead of reissuing new credentials just to extend the lifetime period, Camenisch et al. [4] pointed out that non-interactive credential updates can be a useful replacement. The issuer generates credential update info for all valid credentials before the end of the credential's lifetime is reached. Before the user can send the proof to the verifier, the user needs to download this information and update his credential lifetime. Obviously, a credential to be revoked will not be updated by the issuer and the user will not be able to extend the credential's lifetime.

2.2.1.4.3 Signature Lists

Similar to CRLs in traditional schemes, it is possible to design anonymous revocation lists in anonymous credential systems. However, in order to guarantee the anonymity preservation, the verification of a credential validity results to be more complex. Instead of the verifier performing the verification of the validity of the credential, the user has to prove that the credential he is using is not revoked by the related issuer. This approach, as well as in CRL-based systems, can be provided by following two approaches:

- **Whitelist:** the issuers of the system maintain a list of valid credentials removing from that list the credentials to be revoked. The list consists of signatures on the identifiers of each valid credential and a list identifier. The user selects the signature in the whitelist containing the identifier of his credential and then proves knowledge of the identifier together with the proof that the credential identifier in the signature is the same as the one contained in the credential being validated. Additionally, the list identifier is revealed, such that the verifier can check that the updated (latest) published list was used.
- **Blacklist:** opposite to the whitelist approach, the maintained list comprises all the revoked credentials. Proving non-membership is more complex than proving membership to the whitelist. T. Nakanishi et al. [5] propose an elegant solution by ordering the list of revoked identifiers. For each consecutive pair of identifiers, the issuer publishes a signature on the pair, together with an identifier of the list. During a credential show, the user then proves knowledge of his credential and a signature from the blacklist, such that the identifier in the credential lies between two revoked identifiers in the ordered blacklist. Similar as in the case of whitelists, the disclosed list identifier shows that the latest revocation list was used. If this proof verifies successfully, the verifier is ensured that the credential is valid with respect to the latest published blacklist.

In this approach, for every change that requires the removal of a signature from a whitelist or addition to the blacklist, the issuer has to rebuild the entire revocation list with a new list of identifiers. In case of a join in the whitelist, it is sufficient to add only one signature to the latest whitelist. Likewise, re-approving a previously revoked credential can be done by replacing two consecutive signatures by one new signature. Nevertheless, in both schemes proving membership or non-membership results in a non-negligible, but constant overhead.

2.2.1.4.4 Dynamic Accumulators

A more complex, but possibly more efficient solution for credential revocation is based on the so-called *dynamic accumulators* [6][7][8]. The user needs to prove membership or non-membership in the accumulator. The verifier therefore fetches the latest accumulator value from a revocation authority and if the proof of the credential show verifies correctly w.r.t. that accumulator value, the service provider is ensured that the credential has not been revoked. Except for the verification of a more elaborate proof, the service provider has no additional overhead. On the other hand, although building this proof can be done quite efficiently, it requires the user to first update its witness to enable proving (non-)membership in the accumulator, which is time-consuming. Moreover, since revoking and possibly also adding credentials to the group change the value of the accumulator, a witness update is required. These updates require resources depending linearly to the number of added or revoked credentials from the accumulator.

2.2.1.4.5 ReCRED Credentials Revocation

The ReCRED Attribute Based Access Control infrastructure will provide revocation functionalities for issued credentials considering two possible scenarios distinguished by the required level of privacy. ReCRED will exploit the *Verifiable Encryption* (VE) method to enable revocation of issued credentials in scenarios where the user accepts the eventual disclosure to the issuer of the service provider he accessed (renouncing de-facto to un-traceability properties). In scenarios where it is not possible, since the user cares of un-traceability, ReCRED will exploit the *Limited Lifetime* (LL) with very short validity time of credentials (based on the context) approach that results to be scalable and efficient in terms of performances.

2.2.2 Common Interfaces and Protocols

The Idemix and U-Prove prototypes described below, in Chapter 3, employ common protocols and interfaces that are described in this section. We plan to extend them to enable ABE-based access control as well.

2.2.2.1 FIWARE Privacy Open RESTful API

FIWARE [48] is an open platform that aims to provide a novel service infrastructure to an easier development of future Internet applications. The FIWARE project is the core part of the Future Internet PPP program, a joint initiative by the European Industry and the European Commission.

The open and independent FIWARE community is formed by many individuals and organizations that agree on the FIWARE mission: “to build an open sustainable ecosystem around public, royalty-free and implementation-driven software platform standards that will ease the development of new Smart Applications in multiple sectors”.

The FIWARE platform is the heart of the project and it is mainly focused on technological aspects. Nevertheless, other non-technical relevant activities like FIWARE Lab, FIWARE Accelerator, FIWARE mundus or FIWARE iHubs are important components of the FIWARE ecosystem.

The novel FIWARE infrastructure is built upon basic atomic elements called **generic enablers (GE)**. These are software tools that provide a number of general-purpose functions and are freely available in the rich library of the FIWARE Catalogue. This design makes the development of new services quicker and easier by combining more GE in a modular approach.

Furthermore, each GE is offered through well-defined API (Application Programming Interfaces), easing the development of smart applications in multiple sectors. The set of FIWARE **RESTful API** specifications [49] cover a wide range of different application domains like Cloud Hosting, Internet of Things, security or networks and devices interfaces.

Related to the security area, more security and privacy aspects are in turn taken into consideration and the APIs for Privacy-preserving Authentication in an attribute based infrastructure are well defined too. FIWARE security specifications are based on the ABC4Trust specifications [50] which propose a cryptographic agnostic attribute credential protocol, thus supporting both Idemix and U-Prove. In particular, the FIWARE Privacy GE (Generic Enabler) specifies the API for a P2ABC (Privacy-Preserving Attribute Based Credentials) system.

These APIs describe the endpoints of the main roles that take place in an anonymous authentication system, i.e. Issuers, Users and Verifiers. The APIs are RESTful, resource-oriented, and are accessed via HTTP using XML-based representations for information interchange.

In Table 1, Table 2 and Table 3 the FIWARE APIs for Issuer, User and Verifier are summarized. The tables specify the HTTP method, the URL path, the data given as input inside the body of the request and the returned data in the response. If some parameters can be specified along with the URL path, these are listed above the path.

Table 1 Issuer API

METHOD	PATH	INPUT	OUTPUT	DESCRIPTION
GET	/issuer/setupSystemParameters/	securityLevel	SystemParameters	generates a fresh set of system parameters for the given security level
		cryptoMechanism		
POST	/issuer/setupIssuerParameters/	IssuerParametersInput	IssuerParameters	generates a fresh issuance key and the corresponding issuer parameters
POST	/issuer/initIssuanceProtocol/	IssuancePolicyAndAttributes	IssuanceMessageAndBoolean	Initiate an issuance protocol based on the given issuance policy and the list of attributes to be embedded in the new credential.
POST	/issuer/issuanceProtocolStep/	IssuanceMessage	IssuanceMessageAndBoolean	performs one step in an interactive issuance protocol
GET	/issuer/getIssuanceLogEntry/	issuanceEntryUid	IssuanceLogEntry	looks up an issuance log entry of previously issued credentials

Table 2 User API

METHOD	PATH	INPUT	OUTPUT	DESCRIPTION
POST	/user/canBeSatisfied/	PresentationPolicyAlternatives	ABCEBoolean	on input a presentation policy, decides whether the credentials in the user's credential store could be used to produce a valid presentation token satisfying the policy
POST	/user/createPresentationToken/	PresentationPolicyAlternatives	UiPresentationArguments	returns an argument to be passed to the UI for choosing how to satisfy the policy
POST	/user/createPresentationTokenUi/	UiPresentationReturn	PresentationToken	generates a presentation token that reflects this choice, and which satisfies the respective presentation policy alternatives
POST	/user/issuanceProtocolStep	IssuanceMessage	IssuanceReturn	performs one step in an interactive issuance protocol
POST	/user/issuanceProtocolStepUi/	UiIssuanceReturn	IssuanceMessage	Called after the user has made her choice on how to satisfy the issuance policy
POST	/user/updateNonRevocationEvidence/			updates the non-revocation evidence associated to all credentials in the credential store
GET	/user/listCredentials/		URISet	This method returns an array of all unique credential identifiers (UIDs) available in the Credential Manager

GET	/user/getCredentialDescription/{credentialUid}	credentialUid	CredentialDescription	returns the description of the credential with the given unique identifier.
DELETE	/user/deleteCredential/	credUid	ABCEBoolean	deletes the credential with the given identifier from the credential store

Table 3 verifier API

METHOD	PATH	INPUT	OUTPUT	DESCRIPTION
POST	/verification/verifyTokenAgainstPolicy/	PresentationPolicyAlternatives	PresentationTokenDescription	Checks whether the token satisfies the policy and checks the validity of the cryptographic evidence included in token. Stores the token in a dedicated store if <i>store</i> is set to true
		PresentationToken		
		store		
GET	/verification/getToken/	tokenUID	PresentationToken	looks up a previously verified presentation token
POST	/verification/deleteToken/	tokenUID	Boolean	deletes the previously verified presentation token

2.2.2.1.1 Data format

The relevant data defined in FIWARE and involved in the system setup, issuance and verification phases are summarized above. Moreover, FIWARE specifies an *identity selection* component, involved both in the issuance and verification phases when the user has to choose a preferred combination of credentials and/or pseudonyms.

A more detailed description of the artifacts can be found at [49].

2.2.2.1.2 System parameters

The generic ABCE (ABC Engine) provides 4 setup methods:

- `setupSystemParameters`. This method generates new system parameters (e.g. size of secrets, size of moduli, prime probability, etc.)
- `setupIssuerParameters`. This method generates a secret issuance key and public issuer parameters.
- `setupRevocationAuthorityParameters`. This method generates a secret Revocation Authority key.
- `setupInspectionPublicKey`. This method generates a secret decryption key and a public encryption key for the Inspector.

The credential specification describes the contents of the credentials. It has the following form:

```
<abc:CredentialSpecification Version="1.0" KeyBinding="xs:boolean"
Revocable="xs:boolean">
  <abc:SpecificationUID>xs:anyURI</abc:SpecificationUID>
  <abc:numericalId>xs:integer</abc:numericalId>?
  <abc:FriendlyCredentialName xml:lang="xs:language"/>*
  <abc:DefaultImageReference>xs:anyURI</abc:DefaultImageReference>?
  <abc:AttributeDescriptions MaxLength="xs:unsignedInt">
```

```
<abc:AttributeDescription Type="xs:anyURI" DataType="xs:anyURI"
Encoding="xs:anyURI">
  <abc:FriendlyAttributeName
lang="xs:language">xs:string</abc:FriendlyAttributeName>*
  <abc:AllowedValue>...</abc:AllowedValue>*
</abc:AttributeDescription>*
</abc:AttributeDescriptions>
<abc:CredentialSpecification>
```

The elements presented above describe the following attributes:

```
/abc:CredentialSpecification
```

This element contains the credential content.

```
/abc:CredentialSpecification/@Version
```

This element specifies the version of the credential content.

```
/abc:CredentialSpecification/@KeyBinding
```

This element specifies whether the credential content is bound with a secret key.

```
/abc:CredentialSpecification/@Revocable
```

This element specifies whether the credential content is revocable or not.

```
/abc:CredentialSpecification/SpecificationUID
```

This element contains an identifier for the credential specification.

```
/abc:CredentialSpecification/abc:NumericalId
```

This element contains a numerical identifier for the credential specification.

```
/abc:CredentialSpecification/abc:FriendlyCredentialName
```

This element contains a friendly name for the credential.

```
/abc:CredentialSpecification/abc:FriendlyCredentialName/@lang
```

This attribute contains a localization for the credential.

```
/abc:CredentialSpecification/abc:DefaultImageReference
```

This element contains a reference to the default image for the issued credential.

```
/abc:CredentialSpecification/abc:AttributeDescriptions
```

This element contains the description of the issued attributes.

```
/abc:CredentialSpecification/abc:AttributeDescriptions/abc:AttributeDescription
```

This element contains the description of one attribute.

```
/abc:CredentialSpecification/abc:AttributeDescriptions/abc:AttributeDescription/@Ma
xLength
```

This attribute specifies the maximal length in bits of the integers to which attribute values are mapped using the encoding function

```
/abc:CredentialSpecification/abc:AttributeDescriptions/abc:AttributeDescription/@Ty
pe
```

This attribute contains a unique identifier of an attribute type.

```
/abc:CredentialSpecification/abc:AttributeDescriptions/abc:AttributeDescription/@Da
taType
```

This attribute contains the data type of the credential attribute.

```
/abc:CredentialSpecification/abc:AttributeDescriptions/abc:AttributeDescription/@En
coding
```

This attribute specifies the method for mapping an attribute to an integer value.

```
.../abc:AttributeDescriptions/abc:AttributeDescription/FriendlyAttributeName
```

This element contains a friendly name for the attribute.

```
.../abc:AttributeDescriptions/abc:AttributeDescription/FriendlyAttributeName/@lang
```

This attribute specifies a language identifier for the attribute friendly name.

```
/abc:CredentialSpecification/abc:AttributeDescriptions/abc:AllowedValue
```

This element specifies a list of permitted values for the attribute.

In order for multiple issuers to agree on the cryptographic parameters to use throughout the system, all entities in the system must agree on one set of system parameters. These parameters have to be generated once, before any of the issuer parameters and other keys are generated. The system parameters have the following form:

```
<abc:SystemParameters Version="1.0" SystemParametersUID="xs:anyURI">
  <abc:CryptoParams>...</abc:CryptoParams>
</abc:SystemParameters>
```

The elements described above have the following meaning:

```
/abc:SystemParameters
```

This element contains the system parameters.

```
/abc:SystemParameters/@Version
```

This attribute specifies the system parameters version.

```
/abc:SystemParameters/@SystemParametersUID
```

This attribute specifies a unique identifier for the system parameters.

```
/abc:SystemParameters/abc:CryptoParams
```

This element contains the specific cryptographic elements for the system.

In order to issue credentials, the Issuer must specify system parameters, and generate a key pair consisting of a secret issuing key and a public verification key. The issuer parameters artifact is described below.

```
<abc:IssuerParameters Version="1.0">
  <abc:ParametersUID>xs:anyURI</abc:ParametersUID>
  <abc:FriendlyIssuerDescription
    lang="xs:language">xs:string</abc:FriendlyIssuerDescription>*
  <abc:AlgorithmID>xs:anyURI</abc:AlgorithmID>
  <abc:SystemParametersUID>xs:anyURI</abc:SystemParametersUID>
  <abc:MaximalNumberOfAttributes>xs:int</abc:MaxNumberOfAttributes>
  <abc:HashAlgorithm>xs:anyURI</abc:HashAlgorithm>
  <abc:CryptoParams>...</abc:CryptoParams>
  <abc:RevocationParametersUID>...</abc:RevocationParametersUID>?
</abc:IssuerParameters>
```

The issuer parameters elements presented above describe the following attributes:

```
/abc:IssuerParameters
```

This element contains the issuer public parameters.

```
/abc:IssuerParameters/@Version
```

This attribute contains the version for the issuer parameters.

```
/abc:IssuerParameters/abc:ParametersUID
```

This element contains an identifier for the issuer parameters.

```
/abc:IssuerParameters/abc:FriendlyIssuerDescription
```

This element contains a friendly description for the issuer parameters.

```
/abc:IssuerParameters/abc:FriendlyIssuerDescription/@lang
```

This attribute contains a localization for the issuer parameters friendly name.

```
/abc:IssuerParameters/abc:AlgorithmID
```

This element contains the algorithm for the issuer parameters. The algorithms are Idemix (urn:abc4trust:1.0:algorithm:idemix) and U-Prove (urn:abc4trust:1.0:algorithm:U-Prove).

```
/abc:IssuerParameters/abc:SystemParametersUID
```

This element contains an identifier for the system parameters used with the described issuer parameters.

```
/abc:IssuerParameters/abc:MaximalNumberOfAttributes
```

This element specifies the maximum number of issued attributes.

```
/abc:IssuerParameters/abc:HashAlgorithm
```

This element identifies the hash algorithm which will be used to generate the presentation token.

```
/abc:IssuerParameters/abc:CryptoParams
```

This element contains cryptographic element specific to the employed algorithm.

```
/abc:IssuerParameters/abc:RevocationParametersUID
```

This element contains an identifier for the revocation authority.

In order for the Verifier to communicate to the User which cryptographic algorithms it supports, and provide additional parameters for these algorithms, the verifier must generate a set of verifier parameters and send them to the User. How this artifact is protected (authenticated) is application specific.

```
<abc:VerifierParameters Version="1.0" VerifierParametersId="xs:anyURI"
SystemParametersId="xs:anyURI">
  <abc:CryptoParams>...</abc:CryptoParams>
</abc:VerifierParameters>
```

2.2.2.1.3 Issuance

ABC4Trust and FIWARE propose an identical RESTful protocol message specification. Regarding the attribute token issuance, the user cannot choose or bias the value assigned to the attribute.

The issuer publishes or sends to the User an issuance policy consisting of a presentation policy (which credentials the user must possess in order to be issued an attribute token) and a credential template. The user prepares a special presentation token that fulfills the stated presentation policy, but that contains additional cryptographic information to enable carrying over attribute, user binding, and device binding information.

The User and Issuer subsequently engage in a multi-round issuance protocol, at the end of which the user obtains the requested credential.

ABC4Trust specifies 2 types of issuance: simple issuance (e.g. regular U-Prove issuance) and advanced issuance (the attributes are derived from an existing token). We will describe the simple issuance variant.

The issuance steps are as follows:

1. The user authenticates to the issuer
2. The user specifies which attributes will be issued.
3. The issuer will invoke a generic `initIssuanceProtocol()` method with a set of attributes that shall be certified in the new credential and with an issuance policy that only contains the identifiers of the credential specification and the issuer parameters of the credential that is to be issued. This call initiates an action in the (CE) Crypto Engine (an entity responsible with the underlying crypto implementation – U-Prove). This method returns an issuance message sent to the user.
4. Both sides call a generic `issuanceProtocolStep()` which is called until a credential is issued.

By using such a strategy, the issuance protocol is implementation agnostic.

Any message exchanged in the issuance protocol will be wrapped as an `issuanceMessage`. Because the issuance protocol contains multiple steps, each message includes a Context attributes.

```
<abc:IssuanceMessage Context="...">
</abc:IssuanceMessage>
```

```
/abc:IssuanceMessage
```

This element contains an issuance policy, issuance token or a mechanism specific cryptographic data.

```
/abc:IssuanceMessage/@Context
```

This attribute contains a context for the issuance message.

On the server side, all issued tokens must be logged.

When the issuance protocol is completed, the user obtains a credential which has the type `CredentialDescription`.

```
<abc:CredentialDescription RevokedByIssuer="xs:boolean"?>
  <abc:CredentialUID>
    ...
  </abc:CredentialUID>
  <abc:FriendlyCredentialName lang="xs:language">
    xs:string
  </abc:FriendlyCredentialName>*
  <abc:ImageReference>
    xs:anyURI
  </abc:ImageReference>?
  <abc:CredentialSpecificationUID>
    ...
  </abc:CredentialSpecificationUID>
  <abc:IssuerParametersUID>
    ...
  </abc:IssuerParametersUID>
  <abc:SecretReference>...</abc:SecretReference>?
  <abc:Attribute>
    <abc:AttributeUID>...</abc:AttributeUID>
    <abc:AttributeDescription @Type="xs:anyURI" @DataType="xs:anyURI"
@Encoding="xs:anyURI">
      <abc:FriendlyAttributeName lang="xs:language">
        xs:string
      </abc:FriendlyAttributeName>*
      <abc:AttributeValue>...</abc:AttributeValue>
```

```

        </abc:AttributeDescription>
    </abc:Attribute>*
</abc:CredentialDescription>

```

```
/abc:CredentialDescription
```

This element contains the description of an issued credential.

```
/abc:CredentialDescription/@RevokedByIssuer
```

This attribute contains a flag which states the revocation status for the credential.

```
/abc:CredentialDescription/abc:CredentialUID
```

This element contains a unique identifier for the credential.

```
/abc:CredentialDescription/abc:FriendlyCredentialName
```

This element contains a friendly name for the credential.

```
/abc:CredentialDescription/abc:FriendlyCredentialName/@lang
```

This attribute contains the localization for the friendly name of the credential.

```
/abc:CredentialDescription/abc:ImageReference
```

This element contains a reference to the credential image location.

```
/abc:CredentialDescription/abc:CredentialSpecificationUID
```

This element contains an identifier for the credential specification.

```
/abc:CredentialDescription/abc:IssuerParametersUID
```

This element contains a reference to the issuer parameters.

```
/abc:CredentialDescription/abc:SecretReference
```

This element contains a local identifier for the secret key which is linked with the credential.

```
/abc:CredentialDescription/abc:Attribute
```

This element contains the description of an attribute.

```
/abc:CredentialDescription/abc:Attribute/AttributeUID
```

This element contains a local identifier for the attribute.

```
/abc:CredentialDescription/abc:Attribute/abc:AttributeDescription
```

This element contains the description of an attribute.

```
.../abc:Attribute/abc:AttributeDescription/@Type
```

This attribute contains a unique identifier for the attribute type.

```
.../abc:Attribute/abc:AttributeDescription/@DataType
```

This attribute contains the data type of the attribute.

```
.../abc:Attribute/abc:AttributeDescription/@Encoding
```

This attribute specifies the encoding details of the attribute.

```
/abc:CredentialDescription/abc:Attribute/abc:FriendlyAttributeName
```

This element contains a friendly name for the attribute.

```
.../abc:Attribute/abc:FriendlyAttributeName/@lang
```

This attribute contains a localization for the attribute friendly name.

```
/abc:CredentialDescription/abc:Attribute/abc:AttributeValue
```

This element contains the actual attribute value.

FIWARE specifies that the issuance protocol messages must be wrapped by the Application into a security layer (FIWARE specification mention WS-Trust 1.4 [51]).

The Issuer is responsible for creating the Issuer Parameters and Credential Specifications. FIWARE specifies that the user must use at least one identity source and at least one attribute source (LDAP or JDBC compatible databases).

2.2.2.1.4 Token presentation

To provide certified information to a Verifier entity, the user must present a token that contains a series of attributes or statements regarding her credentials. Beside from revealing information about user credentials, the token can be used to sign messages, in order to ensure freshness.

The token must support paradigms like: pseudonyms, key binding, inspection and revocation. The Verifier can cryptographically verify the authenticity of a received presentation token using the credential specifications and issuer parameters of all credentials involved in the token. The Verifier must obtain the credential specifications and issuer parameters in a trusted manner, e.g., by using a traditional PKI to authenticate them or retrieving them from a trusted location. The process of presentation is triggered when the application on the user's side contacts a verifier to request access to a resource.

The presentation steps are as follows:

1. Presentation is triggered when user wants to access a protected resource
2. The Verifier responds with one or more presentation policies. It may specify which credentials from which trusted issuer are required, which attributes must be revealed etc.
3. The user invokes a generic method `createPresentationToken()`. The token is sent to the verifier.
4. The verifier calls a generic method `verifyTokenAgainstPolicy()`. This method verifies the token statements according to the presentation policy. If the verification succeeds, the token is stored.

The presentation policy sent by the Verifier has the following schema:

```
<abc:PresentationPolicyAlternatives Version="1.0">
<abc:PresentationPolicy PolicyUID="xs:anyURI"?>
  <abc:Message>
    <abc:Nonce>...</abc:Nonce>?
    <abc:FriendlyPolicyName lang="xs:language">
      xs:string
    </abc:FriendlyPolicyName>*
    <abc:FriendlyPolicyDescription lang="xs:language">
      xs:string
    </abc:FriendlyPolicyDescription>*
    <abc:VerifierIdentity>
      xs:any
    </abc:VerifierIdentity>?
    <abc:ApplicationData>
      ...
    </abc:ApplicationData>?
  </abc:Message>?
</abc:PresentationPolicy>
<abc:Pseudonym Exclusive="xs:boolean"? Scope="xs:string"
```

```

Established="xs:boolean"? Alias="xs:anyURI"? SameKeyBindingAs="xs:anyURI"?
  <abc:PseudonymValue> </abc:PseudonymValue>?
</abc:Pseudonym>*
<abc:Credential Alias="xs:anyURI"? SameKeyBindingAs="xs:anyURI"?
  <abc:CredentialSpecAlternatives>
    <abc:CredentialSpecUID>...</abc:CredentialSpecUID>+
  </abc:CredentialSpecAlternatives>
  <abc:IssuerAlternatives>
    <abc:IssuerParametersUID RevocationInformationUID="xs:anyURI"?
      ...
    </abc:IssuerParametersUID>+
  </abc:IssuerAlternatives>
  <abc:DisclosedAttribute AttributeType="xs:anyURI"
DataHandlingPolicy="xs:anyURI"?
    (
      <abc:InspectorAlternatives>
        <abc:InspectorPublicKeyUID>
          ...
        </abc:InspectorPublicKeyUID>+
      </abc:InspectorAlternatives>
      <abc:InspectionGrounds>
        ...
      </abc:InspectionGrounds>
    )?
  </abc:DisclosedAttribute>*
</abc:Credential>*
<abc:VerifierDrivenRevocation>
  <abc:RevocationParametersUID>...</abc:RevocationParametersUID>
  <abc:AttributeCredentialAlias="xs:anyURI" AttributeType="xs:anyURI">+
</abc:VerifierDrivenRevocation>*
<abc:AttributePredicate Function="xs:anyURI">
  (
    <abc:Attribute CredentialAlias="xs:anyURI" AttributeType="xs:anyURI"
DataHandlingPolicy="xs:anyURI"?/>
    |
    <abc:ConstantValue>...</abc:ConstantValue>
  )+
</abc:AttributePredicate>*
</abc:PresentationPolicy>+
</abc:PresentationPolicyAlternatives>

```

The presentation of one or multiple credentials results in a presentation token that is sent to the verifier. The syntax for the element is:

```

<abc:PresentationToken Version="1.0">
  <abc:PresentationTokenDescription PolicyUID="xs:anyURI"
TokenUID="xs:anyURI"?
    <abc:Message>
      <abc:Nonce>...</abc:Nonce>?
      <abc:FriendlyPolicyName lang="xs:language">
        xs:string
      </abc:FriendlyPolicyName>*
      <abc:FriendlyPolicyDescription lang="xs:language">
        xs:string
      </abc:FriendlyPolicyDescription>*
      <abc:VerifierIdentity>xs:any</abc:VerifierIdentity>
      <abc:ApplicationData>...</abc:ApplicationData>?
    </abc:Message>?
    <abc:Pseudonym Scope="xs:string"? Exclusive="xs:boolean"?
Alias="xs:anyURI"? SameKeyBindingAs="xs:anyURI"?
      <abc:PseudonymValue>...</abc:PseudonymValue>
    </abc:Pseudonym>*
    <abc:Credential Alias="xs:anyURI"? SameKeyBindingAs="xs:anyURI"?
      <abc:CredentialSpecUID>...</abc:CredentialSpecUID>
      <abc:IssuerParametersUID>...</abc:IssuerParametersUID>
      <abc:RevocationInformationUID>
        ...
      </abc:RevocationInformationUID>?
      <abc:DisclosedAttribute AttributeType="xs:anyURI"
DataHandlingPolicy="xs:anyURI"?

```

```

(
  <abc:InspectorPublicKeyUID>...</abc:InspectorPublicKeyUID>
  <abc:InspectionGrounds>...</abc:InspectionGrounds>
) ?
  <abc:AttributeValue>...</abc:AttributeValue>
  </abc:DisclosedAttribute>
</abc:Credential>*
<abc:VerifierDrivenRevocation>

  <abc:RevocationInformationUID>...</abc:RevocationInformationUID>
  <abc:Attribute AttributeType="xs:anyURI" CredentialAlias="
xs:anyURI" >+
    </abc:VerifierDrivenRevocation>*
    <abc:AttributePredicate Function="xs:anyURI">
      (
        <abc:Attribute CredentialAlias="xs:anyURI"
        AttributeType="xs:anyURI" DataHandlingPolicy="xs:anyURI"?/>
      |
    <abc:ConstantValue>...</abc:ConstantValue>
      )+
    </abc:AttributePredicate>*
</abc:PresentationTokenDescription>
<abc:CryptoEvidence>
...
</abc:CryptoEvidence>
</abc:PresentationToken>

```

```
/abc:PresentationToken
```

This element contains a presentation token.

```
/abc:PresentationToken/@Version
```

This attribute contains the presentation token version.

```
/abc:PresentationTokenDescription
```

This element contains the description for the disclosed attributes.

```
.../abc:PresentationPolicy/@PolicyUID
```

This attribute contains an identifier for the presentation policy.

```
.../abc:PresentationPolicy/@TokenUID
```

This attribute contains a unique identifier for the presentation token.

```
.../abc:PresentationTokenDescription/abc:Message
```

This element contains a signed message by private key of each cryptographic token (credential).

```
.../abc:PresentationTokenDescription/abc:Message/abc:Nonce
```

This element contains a random number signed by the private key of each token.

```
.../abc:PresentationTokenDescription/abc:Message/abc:FriendlyPolicyName
```

This element contains a friendly name for the description.

```
.../abc:PresentationTokenDescription/abc:Message/abc:FriendlyPolicyName/@lang
```

This attribute specifies the localization for the friendly name policy.

```
.../abc:PresentationTokenDescription/abc:Message/abc:VerifierIdentity
```

This element contains the identity of the verifier.

```
.../abc:PresentationTokenDescription/abc:Message/abc:FriendlyPolicyDescription
```

This element contains a friendly description for the policy.

```
.../abc:Message/abc:FriendlyPolicyDescription/@lang
```

This attribute contains a localization for the friendly description of the policy.

```
.../abc:PresentationTokenDescription/abc:Message/abc:ApplicationData
```

This element contains the data type of the string.

```
.../abc:PresentationTokenDescription/abc:Pseudonym
```

This element indicates that a pseudonym is present along with the presentation token.

```
.../abc:PresentationTokenDescription/abc:Pseudonym/@Scope
```

This attribute contains the scope of the pseudonym.

```
.../abc:PresentationTokenDescription/abc:Pseudonym/@Exclusive
```

This attribute indicates that the pseudonym is scope-exclusive.

```
.../abc:PresentationTokenDescription/abc:Pseudonym/@Alias
```

This attribute contains an alias for the pseudonym.

```
.../abc:PresentationTokenDescription/abc:Pseudonym/@SameKeyBindingAs
```

This attribute contains an alias to another pseudonym or to a Credential element for a credential key binding.

```
.../abc:PresentationTokenDescription/abc:Pseudonym/abc:PseudonymValue
```

This element contains the base64 encoding of the pseudonym.

```
.../abc:PresentationTokenDescription/abc:Credential
```

This element contains the presentation token credential.

```
.../abc:PresentationTokenDescription/abc:Credential/@Alias
```

This attribute contains an alias for the credential.

```
.../abc:PresentationTokenDescription/abc:Credential/@SameKeyBindingAs
```

This attribute contains an alias for a pseudonym or a reference to another Credential element for credential with key binding.

```
.../abc:Credential/abc:CredentialSpecUID
```

This element contains an identifier for the credential.

```
.../abc:PresentationTokenDescription/abc:Credential/abc:IssuerParametersUID
```

This element contains an identifier for the credential public key.

```
.../abc:PresentationTokenDescription/abc:Credential/abc:RevocationInformationUID
```

This element contains an identifier for the revocation information.

```
.../abc:PresentationTokenDescription/abc:Credential/abc:Attributes
```

This element contains the disclosed attributes.

```
.../abc:PresentationTokenDescription/abc:Credential/abc:DisclosedAttribute
```

This element contains one disclosed attribute.

```
.../abc:Credential/abc:DisclosedAttribute/@AttributeType
```

This element describes the credential type of the disclosed attribute.

```
.../abc:Credential/abc:DisclosedAttribute/@DataHandlingPolicy
```

This attribute contains an external data handler policy.

```
.../abc:Credential/abc:DisclosedAttribute/abc:InspectorPublicKeyUID
```

This optional element contains the identifier of the inspector public key under which the attribute value is encrypted

```
.../abc:Credential/abc:DisclosedAttribute/abc:InspectionGrounds
```

This element contains the context under which the inspector can decrypt the attributes.

```
.../abc:Credential/abc:DisclosedAttribute/abc:AttributeValue
```

This element contains the base64 encoding of the disclosed attribute.

```
.../abc:PresentationTokenDescription/abc:VerifierDrivenRevocation
```

The element describes the parameters used in the validity verification of an attribute.

```
.../abc:PresentationTokenDescription/abc:VerifierDrivenRevocation/abc:RevocationInformationUID
```

This element contains an identifier for the verification information.

```
.../abc:PresentationTokenDescription/abc:VerifierDrivenRevocation/abc:Attribute
```

This element specifies a credential attribute that is used for verifier-driven revocation

```
.../abc:PresentationTokenDescription/abc:VerifierDrivenRevocation/abc:Attribute/@CredentialAlias
```

This attribute describes an alias for the credential from which the attribute was used.

```
.../abc:PresentationTokenDescription/abc:VerifierDrivenRevocation/abc:Attribute/@AttributeType
```

This attribute refers to the attribute used for verifier-driven information.

```
.../abc:PresentationTokenDescription/abc:AttributePredicate
```

This optional element specifies a predicate that is guaranteed to hold by this token.

```
.../abc:AttributePredicate/@Function
```

This attribute specifies the boolean function for this predicate.

```
.../abc:AttributePredicate/abc:Attribute
```

This element contains a reference to the attribute used in the predicate evaluation.

```
.../abc:AttributePredicate/abc:Attribute/@CredentialAlias
```

This attribute contains an alias for the credential which contains the attribute.

```
.../abc:AttributePredicate/abc:Attribute/@AttributeType
```

This attribute contains the exact attribute used as a predicate argument.

```
.../abc:AttributePredicate/abc:Attribute/@DataHandlingPolicy
```

This attribute contains an external data handling policy used in predicate evaluation.

```
.../abc:AttributePredicate/abc:ConstantValue
```

This element specifies a constant value used for the predicate.

```
/abc:PresentationToken/abc:CryptoEvidence
```

This element contains the cryptographic elements for the presentation token.

2.2.2.1.5 Identity Selection

The Identity Selection steps are executed between the user and the identity selection component. These allow the user to choose among different combination of credentials and/or pseudonyms in order to satisfy a presentation or an issuance policy. The identity selection component can be considered as user interface (UI), for instance a graphic interface. More details can be found in the FIWARE specification [48].

2.2.2.2 IRMA (*I Reveal My Attributes*)

The IRMA (I Reveal My Attributes) project [9], aims at providing an open, secure, decentralized and easy to use implementation of Privacy-Preserving Attribute-Based Access Control with minimal disclosure of attributes for online and offline transactions.

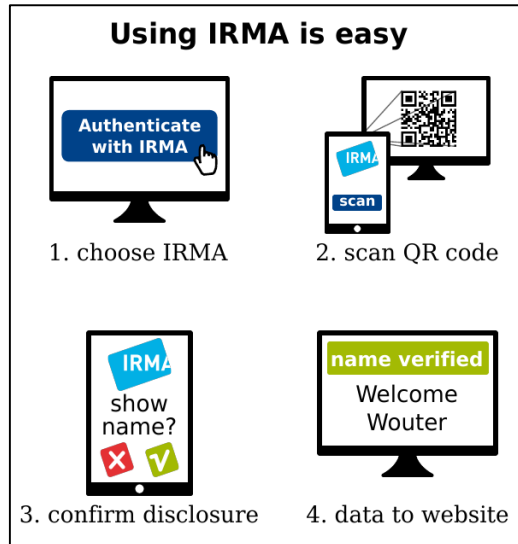


Figure 7 "Using IRMA is easy" - from the IRMA Project [9]

From the point of view of the user, she installs the IRMA mobile app, which acts as an Attribute-Based Credential wallet. Then, with her desktop or laptop, she goes to the website of an issuer, authenticates with it and requests the issuance of a credential. The website shows a QR code and the user scans it with the IRMA app. The issued credential is transferred to the user device and, if the user confirms the transaction, stored in the wallet.

Then, when the user visits a website and requests access to a protected resource (e.g. a video on IRMATube [10]), the website shows a QR code. When the user scans the QR code with the IRMA app, she is prompted with the attributes that will be disclosed to the website and asked for confirmation. If the user confirms, the website presents to the user the protected resource. The simplified data flow for the IRMA verification process is depicted in Figure 8.

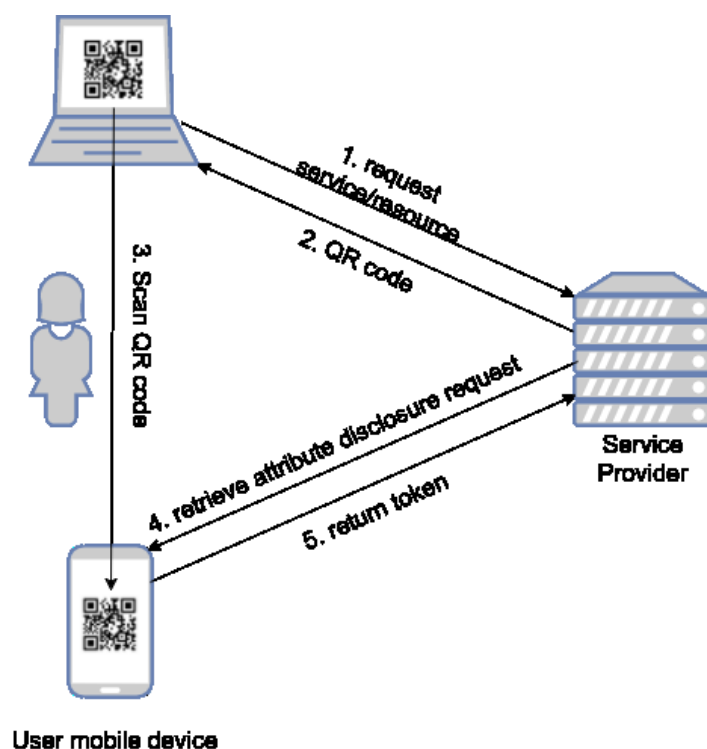


Figure 8 IRMA data flow (verification)

For ReCRED, the open source Idemix implementation targeted at mobile devices of IRMA is particularly relevant: its ease of use and its device-centric and user-centric approach are well aligned with the objectives of ReCRED. Thus, we are currently using it as the underlying Idemix crypto engine for the ReCRED ABAC components, and is the basis of the ReCRED Wallet application, described in Section 3.1.1.2.2.

2.2.2.2.1 IRMA Architecture

We here provide a description of the architecture of the IRMA Idemix implementation targeted at mobile devices. The main components, the User Device, the IRMA API Server and the Application Server are depicted in Figure 9 and described below. Please note that with respect to Figure 8 the Service Provider side is split into two components: the IRMA API Server, which performs the cryptographic Idemix operations, and the Application Server, which runs the actual application logic.

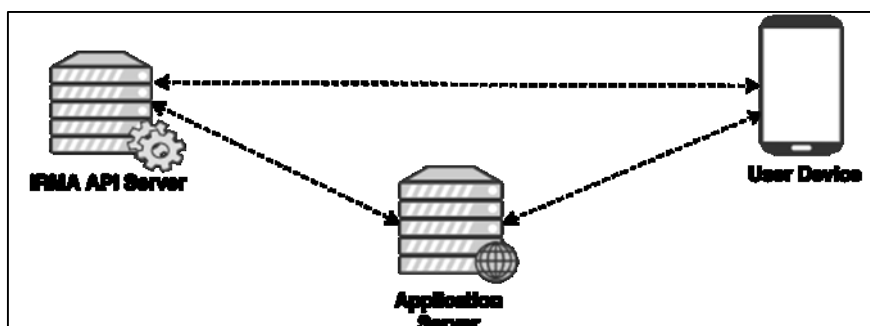


Figure 9 IRMA for Mobile Devices Architecture

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

The design of the IRMA architecture allows to have a cryptographic implementation clearly separated with respect to the service/identity-provider specific application level.

2.2.2.2.2 IRMA Issuance

The diagram in Figure 10 summarizes the issuance process in the IRMA implementation targeted at mobile devices. When the user requests a credential to an Identity Provider, the Application Server requests the IRMA API Server to generate a session token. This session token is passed to the User Device through e.g. a QR code. The User Device uses this session token to contact the IRMA API Server and retrieve the attributes that will be included in the issued credential. If the End User gives her consent, the requested Idemix credential is issued to the User Device and the Identity Provider is notified.

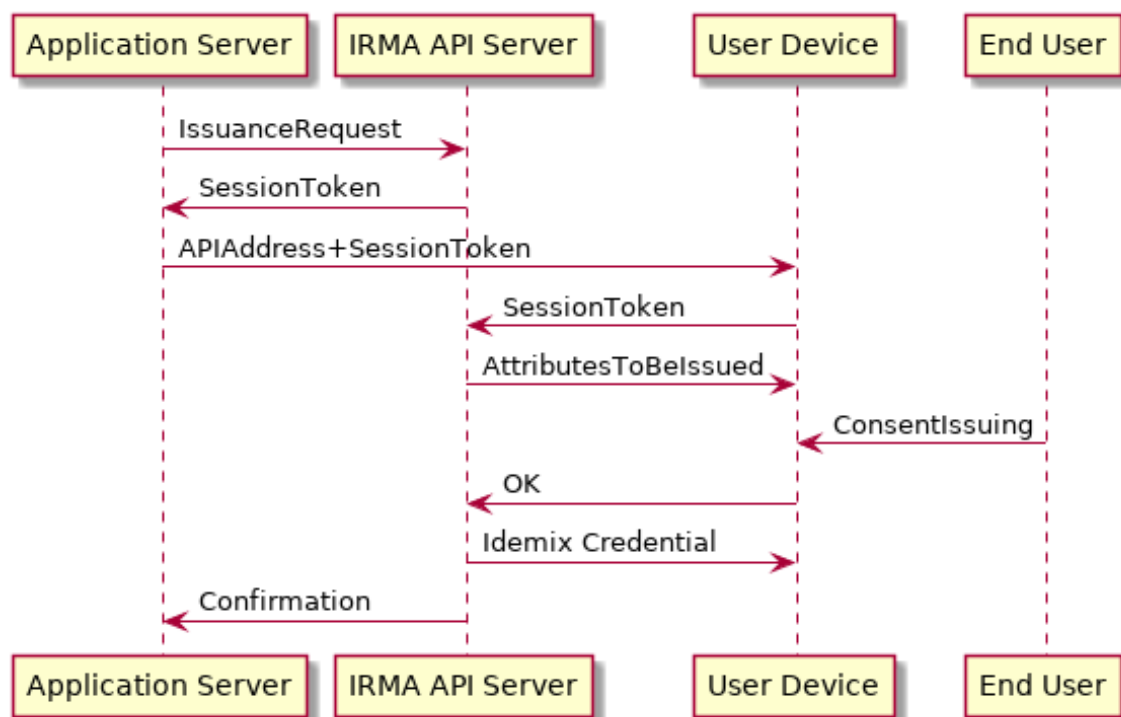


Figure 10 IRMA Issuance sequence diagram

2.2.2.2.3 IRMA Verification

The diagram in Figure 11 depicts the verification process in the IRMA implementation targeted at mobile devices. When the user requests a resource to a service provider, the Application Server requests the IRMA API Server to generate a session token. This session token is passed to the User Device through e.g. a QR code. The User Device uses this session token to contact the IRMA API Server and retrieve the list of attributes which are required to access the resource. If the End User consents

to the disclosure of the required attributes, the User Device proves to the IRMA API Server the possession of the required attributes. Alternatively, the User can prove only the possession of a specific credential (without revealing the included attributes). The IRMA API Server verifies the validity of the received proofs and sends the required attributes to the Application Server. The Application Server then sends the requested resource to the user.

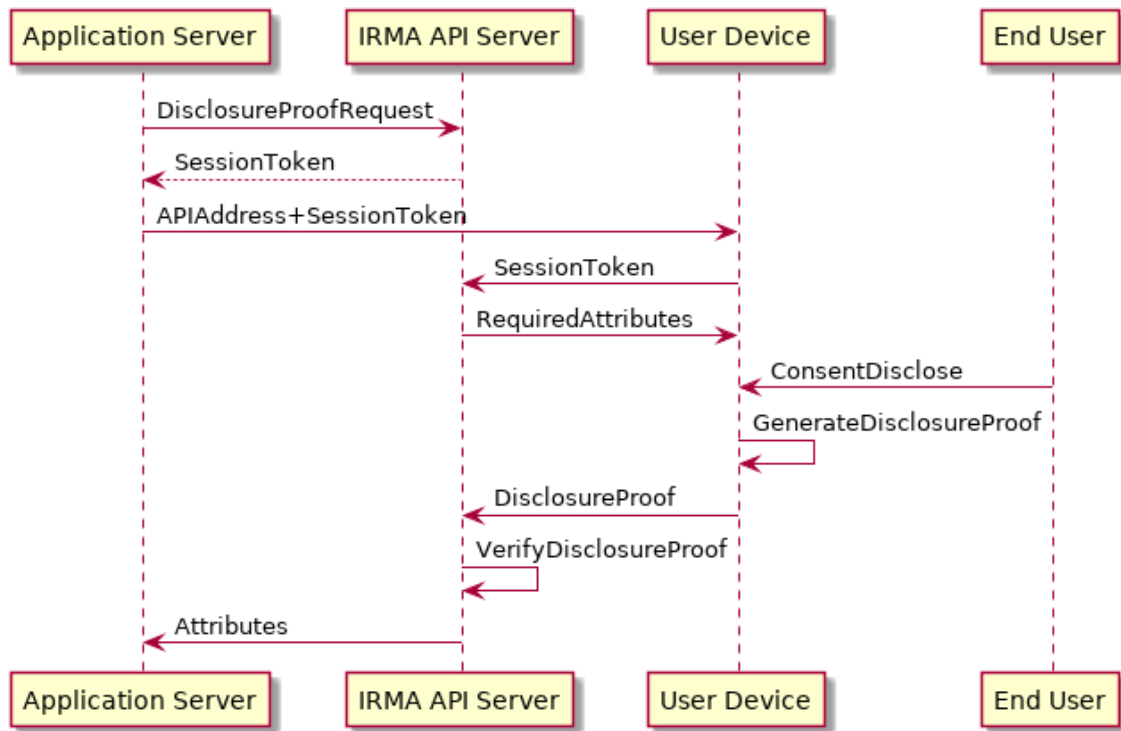


Figure 11 IRMA Verification Sequence Diagram

2.2.2.3 Consent Management Design

Consent Management consists of the following sub-modules:

- The Consent Management **front-ends**, which provide a UI so that the ReCRED users and Identity Providers can view and manage their consent policies.
- **REST APIs** that provide CRUD functionality, as well as consent policy enforcement.
- The Consent Management **data store**, for persistently storing the consent policies of the ReCRED users and the Identity Providers.

The following diagram (Figure 12) depicts the main sub-modules of the Consent Management Module and their relations among them and with other ReCRED modules.

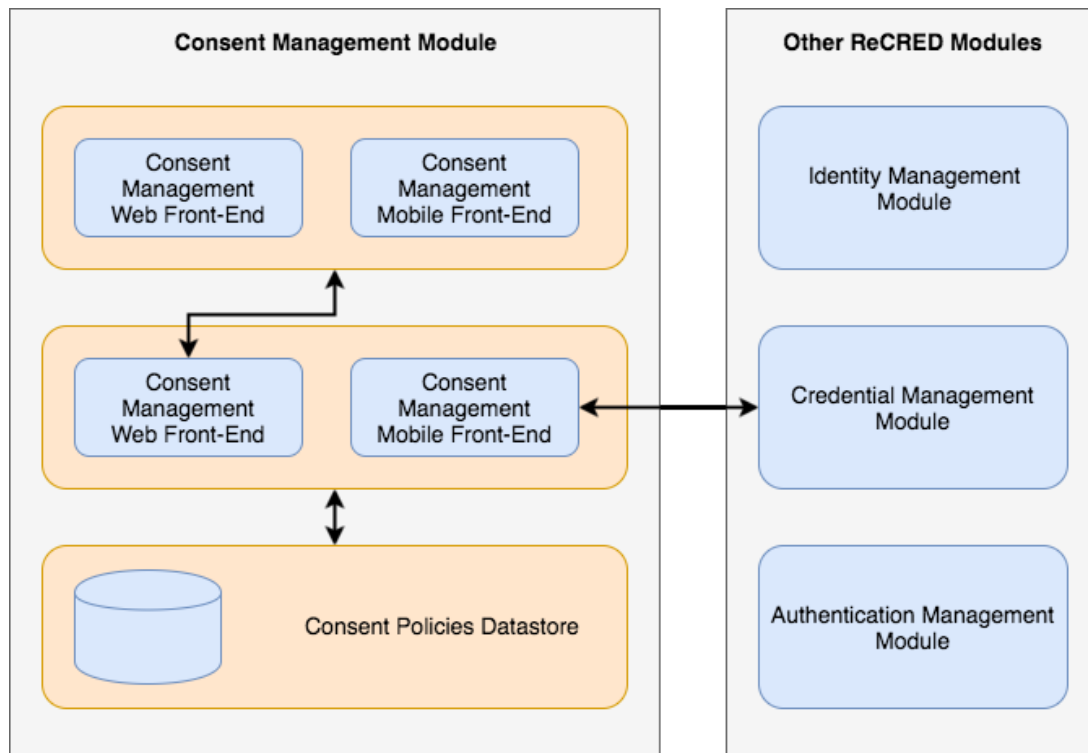


Figure 12: Consent Management Design

Two front-ends are provided, through which the ReCRED users and the Identity Providers can manage their consent policies. The **Consent Management Web Front-end** is accessible through all the major web browsers, and it allows authorized end-users and Identity Providers to create new consent policies, see the policies that they have created and edit or delete them. The ReCRED users can create policies, according to which specific identity attributes will be hidden to specific Identity and/or Service Providers. The Identity Providers can additionally create policies, according to which the issuance of cryptographic credentials (Idemix / U-Prove) will be forbidden for specific attributes. The **Consent Management Mobile Front-end** is an Android app, which provides the same functionality but only for the ReCRED end-users.

The **Consent Management API** exposes all the required CRUD operations for managing the consent policies. HTTP POST is used, in order to create and save new policies to the data store, HTTP GET is used to retrieve the user’s policies, and HTTP PUT and DELETE are used in order to update and delete consent policies respectively.

The **Policy Enforcement Engine (PEE)** also uses REST calls, in order to expose operations for the enforcement of the consent policies of the ReCRED users and the Identity Providers. This API is used by other ReCRED modules, which need to abide by these policies. More specifically:

- The **Identity Management Module** calls the PEE each time a user attempts to transfer an identity attribute between two IdPs (A -> B). In that case, the PEE ensures that there are no policies, defined either by this user or by the source IdP (A), according to which the specific identity attribute should not be revealed to the target IdP (B). If there are such policies, the PEE blocks the attribute transfer, otherwise the transfer is approved.

- The **Credential Management Module** calls the PEE each time it attempts to issue an idemix / u-prove credential to the user’s device, including specific identity attributes. In that case, the PEE ensures that there are no policies defined by the IdP, forbidding one or more of these attributes to be proven using Idemix / U-Prove. If there are such policies, the PEE blocks the credential issuance, otherwise the issuance is approved.
- The **OpenAM** module calls the PEE each time it attempts to reveal a user’s identity attribute to a Service Provider. In that case, the PEE ensures that there are no policies, defined either by this user or by the IdP holding the attribute value, according to which the specific identity attribute should not be revealed to the specific Service Provider. If there are such policies, the PEE blocks the attribute reveal, otherwise the reveal is approved.

Finally, the **Consent Policies Data Store** sits at the bottom of the architecture and stores all the consent policies created by the ReCRED users and the Identity Providers.

3 P-ABAC Module Implementation and Mapping to P-ABAC components

This section reports on the implementation of the modules that support the ReCRED P-ABAC infrastructure. After the description of the implementation of the modules, we map them to the ReCRED ABAC components presented in Section 2.1.1.

3.1 P-ABAC Components Implementation

3.1.1 Credential Management Daemon

One of the most important features provided by the ReCRED framework is the implementation of a user-friendly Attribute Based Access Control (ABAC) architecture, integrated in the system, allowing the user to preserve the privacy of its attributes.

The Credential Management Daemon is designed to “translate” identity attributes acquired by Identity Providers to valid credentials for the Privacy-Preserving ABAC (P-ABAC) cryptographic technologies adopted by ReCRED (i.e. Idemix, U-Prove, ABE).

The Credential Management Module of the ReCRED Identity Consolidator is a special instance of the Credential Management Daemon: it is in charge of issuing credentials to the user, compiling the credential attribute values from verified identity attributes stored by the Identity Consolidator. These values are retrieved by the Credential Management Module by using the Storage Module API.

3.1.1.1 P-ABAC API Server

The ReCRED P-ABAC API Server is the server demanded to run the cryptographic functionalities for Idemix and U-prove. The following sections provide an overview of the issuance protocol performed when issuing Idemix and U-prove credentials.

3.1.1.1.1 Idemix API Server

The issuing protocol is triggered when a user requires the issuance of Idemix credentials from an Identity Provider.

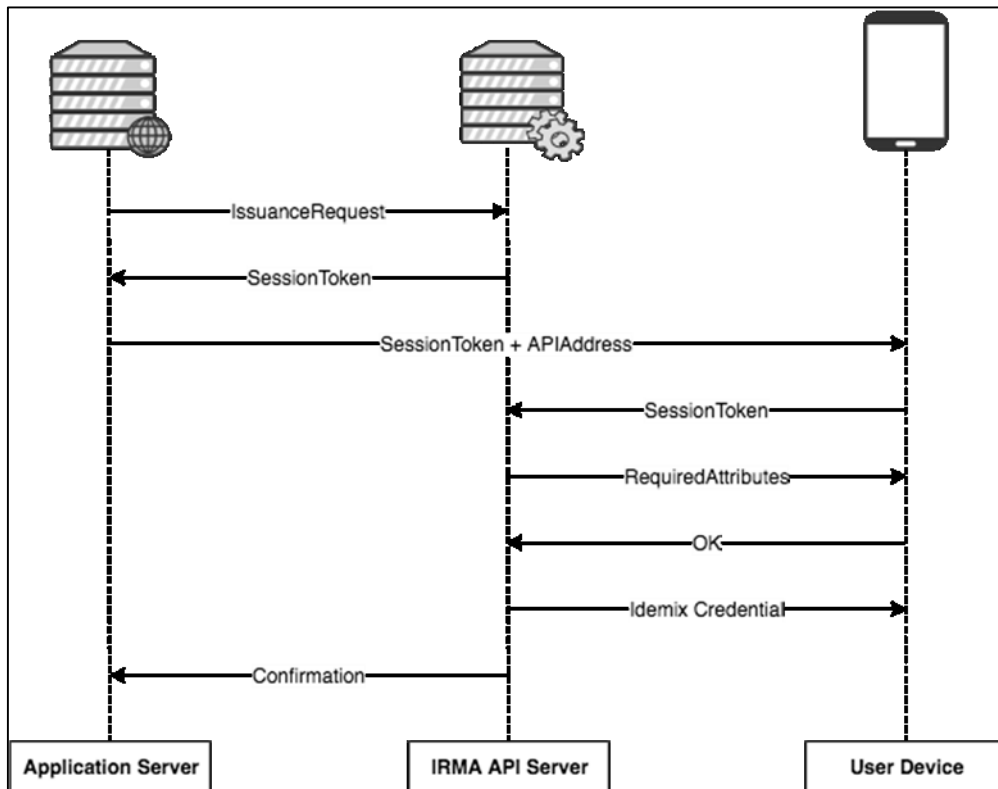


Figure 13. IRMA Issuance Protocol

The issuance protocol’s phases reported in Figure 13 are the following:

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

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

3.1.1.1.2 U-Prove API Server

The U-Prove API service contains a U-Prove engine which performs all the cryptographic operations during the issuance protocol. For the communication between client and server the U-Prove objects are serialized using the JSON format. The following image illustrates how the data is being sent over the network from the client to the server.

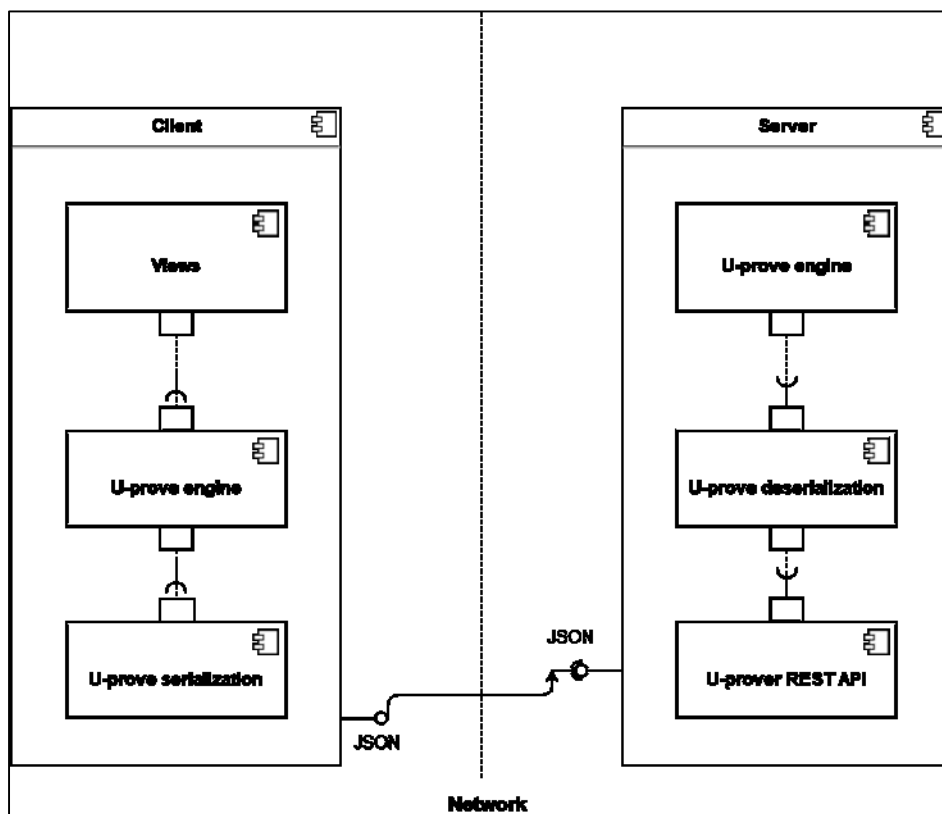


Figure 14. U-Prove client-server architecture

The output of the issuance process is a U-Prove token containing the following information:

1. An application-specific unique identifier for the Issuer parameters under which this token was issued;
2. the *Token Information* field is used to encode token-specific information which is always disclosed to Verifiers, such as token usage restrictions, a validity period, or any other metadata. If this information is significant then the implementation should provide the particular implementation for the decoding process in the verification process;
3. the *Prover Information* field is used to encode token-specific information which is always disclosed to Verifiers, such as token usage restrictions, a validity period, or any other metadata. This information is also disclosed to the Verifier;
4. The signature provided by the Issuer.

For keeping track of all the U-Prove clients that are accessing the U-Prove server, the API generates a session key for each client that tries to obtain a U-Prove credential. When a user tries to generate multiple U-Prove credentials (tokens) all the following is being verified:

- the U-Prove session key which the user presents exists in the database and the session key that the user provided is not expired;

- the U-Prove client together with the valid session key previously validated makes a valid request to the U-Prove server. This aspect must be checked because the issuance process in U-Prove takes more than one step;
- the identity of the issuer doesn’t exist in different states at the same time during the issuance process. More exactly the same issuer should not generate a U-Prove credential for more than one user at the same time;
- the number of credentials the user wants to obtain is smaller than the maximum number of credentials the server can provider at once.

3.1.1.2 Credential Management Application

One of the aims of ReCRED is to support an Attribute Based Access Control Infrastructure that enables the use of anonymous credentials, based on technology at the state of the art such as Idemix and U-Prove. These technologies provide the core cryptographic functionalities of the P-ABAC architecture but have different requirements for what concerns their interfaces. The FIWARE project provides a common interface between Idemix and U-Prove by means of the Privacy Generic Enabler [4]. In the ReCRED ABAC architecture we aim at joining together the ReCRED results with the outputs from the IRMA and FIWARE projects to provide an integrated P-ABAC architecture, providing both Idemix and U-Prove credentials support.

The Credential Management (CM) module aims at supporting the issuance of anonymous, P-ABAC credentials derived from the information acquired by the Identity Consolidator. The P-ABAC credentials are issued to the User Device and received and stored by the ReCRED Wallet app.

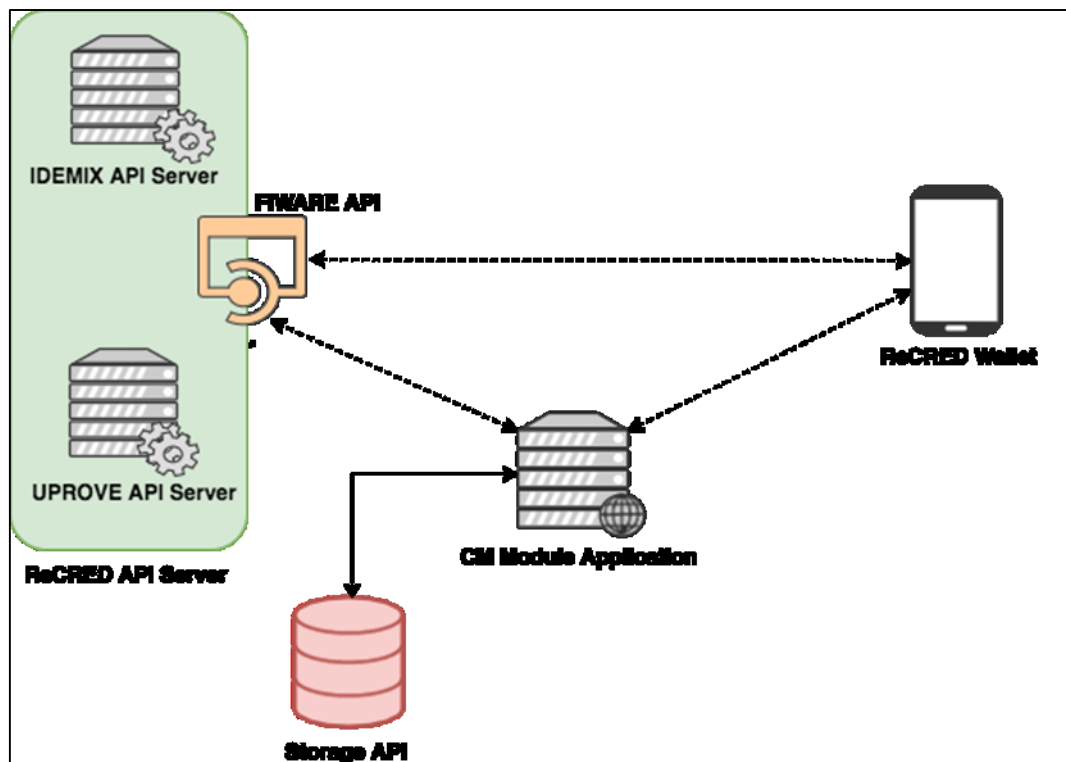


Figure 15. Credential Management module submodules and interactions

As shown in figure 12, the Credential Management Module is composed by two main sub-components reflecting the Issuer component structure:

- **ReCRED API Server**
- **Credential Management Module Application**

3.1.1.2.1 Credential Management Module Application Frontend

The Credential Management Module frontend is a web application that can be used by the user to list the credentials that can be issued to her by the Identity Consolidator. The application is linked to the Storage API, in order to be able to retrieve identity attributes to be used in the credential generation, and to the ReCRED API Server, for the actual credentials issuance.

We provide below a list of operations which can be performed by the user on the Credential Management Module frontend.

3.1.1.2.1.1 List Compiled Credentials

Once the user logs into the CM Module Application Frontend, the application retrieves through the Storage API all attribute values that match the fields of a set of pre-defined credentials. These attribute values are used to fill the fields of these credential structures, i.e. to compile the credentials to be issued. These credentials are listed to the user as shown in Figure 16. For each credential the following information is presented:

- **Credential Name:** is the name of the credential represented in the following dotted format:
 - **<Domain>.<Issuer>.<CredentialName>** where:
 - **Domain:** is the application domain, which in the ReCRED project case will be **recred**;
 - **Issuer:** is the issuer name, which in the case of the CM Module is **RecredIdentityConsolidator**, since the issuer is the Identity Consolidator;
 - **CredentialName:** is the name that identifies the credential (e.g. usernames, userphone)
- **List of attributes:** is the list of attributes defined in the credential structure and for which the value has been retrieved through the Storage API.

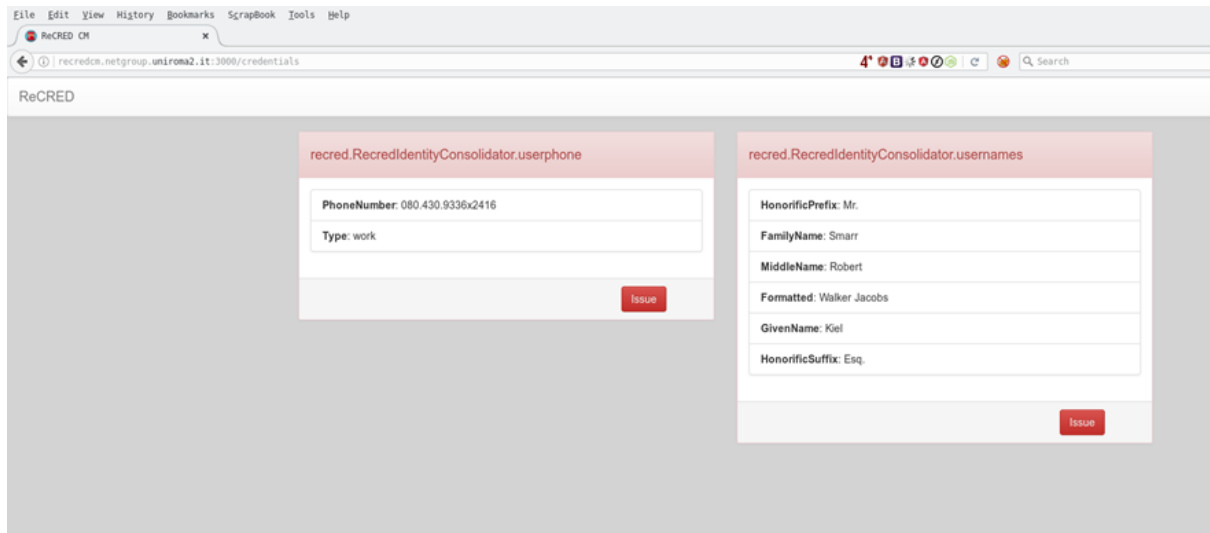


Figure 16. List of available credential ready to be issued

3.1.1.2.1.2 Issue Credentials

From the credentials list, the user is able to select the credential that wants to be issued to her own device. Once it selects the “Issue” button for a specific credential, the CM Module Application contacts the ReCRED API Server to obtain a session token for the issuance of the selected credential. Once the Credential Management Module application receives the session token from the API Server, it provides to the user a QR Code, as shown in Figure 17 that contains a JSON text with the following information:

- **Session Token**, e.g., eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9
- **URL of the API Server**: e.g., <https://api.recred.eu/abac>
- **Protocol Version**: e.g., v1

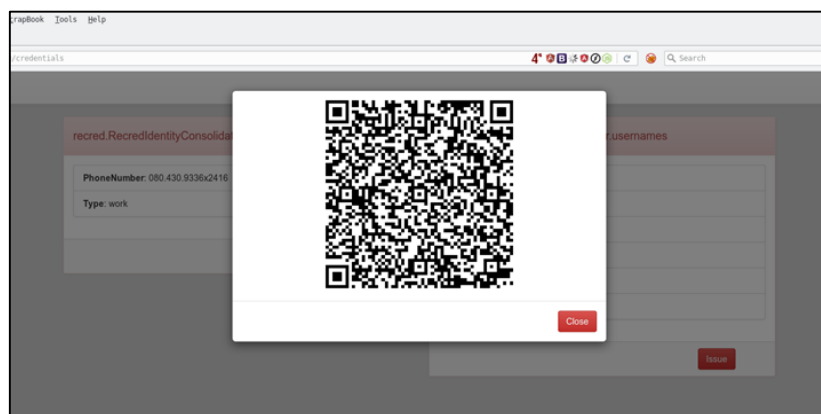


Figure 17. Credential Issuing: Link of the User Device to the API server by means of QR Code

The QR code enables the user to receive the information about the opened session with the API Server in order to execute the Issuance protocol directly with the API Server itself.

3.1.1.2.2 Wallet Application

The ReCRED Wallet Application is designed both to manage P-ABAC credentials and support the user in the Issuance and Verification phases. Indeed, the user is able to receive the credentials directly to the user device thanks to such application. Moreover, all the issued credentials will be available to the user through the Wallet Application. By using the ReCRED Wallet Application, shown in Figure 18, the user can scan the QR code and receive such information.

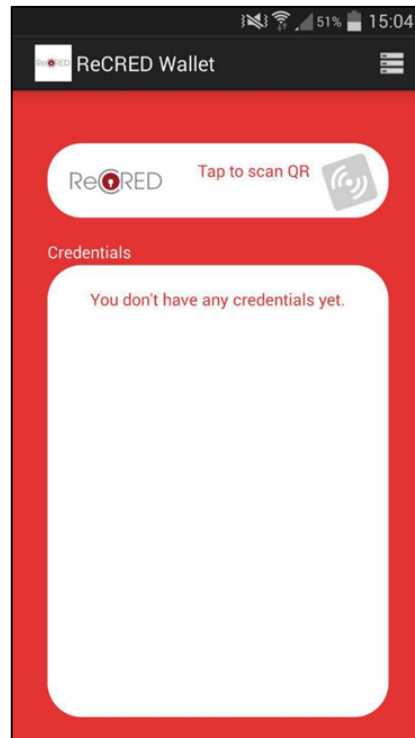


Figure 18. ReCRED Wallet Application

At this point the ReCRED Wallet Application and the API Server will run the Issuance Protocol in order to perform the issuance of the previously selected credential. As shown in Figure 19, the user will be asked to provide consent to receive the credential and the attributes, together with the value of each attribute, displayed to the user by means of a dialog. The user can deny consent, and the issuance of the credential will be stopped. Otherwise the credential will be issued to the user and will be stored in the user device itself.

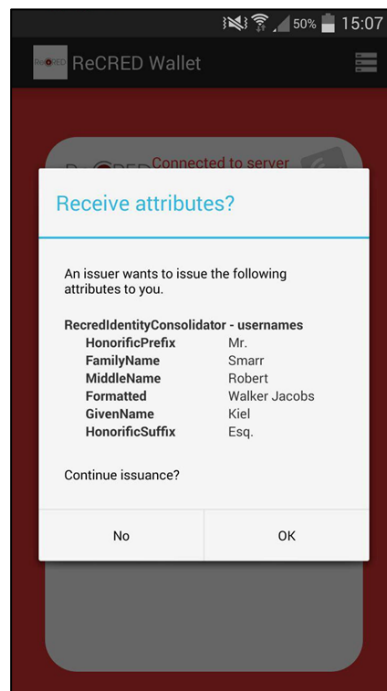


Figure 19. ReCRED Wallet Application: issuance consent display

Once the issuance of the credential is completed, it is added to the list of credentials owned by the user and issued to her device (Figure 20).

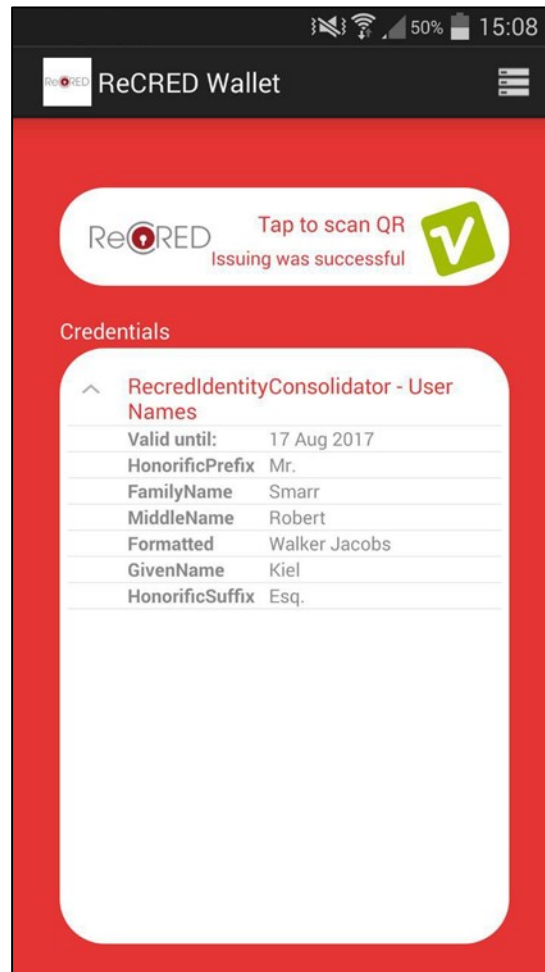


Figure 20. ReCRED Wallet application: Credential list and details

3.1.1.3 Integration of the Credential Management in the Identity Consolidator

The Credential Management Daemon has been integrated in the ReCRED Identity Consolidator. To this aim a docker-compose file and a set of Docker files have been produced, corresponding to the Credential Management Frontend, the Credential Management backend, the API Server configuration and the actual API crypto Server.

3.1.2 U-Prove Implementation

The implementation of U-Prove was written completely in the Java language in order to provide interoperability. The code is focused around two major components:

- a U-Prove engine which is responsible for all the cryptographic operations and implements the U-Prove protocol specifications;
- a U-Prove REST API used primarily for integration with the other components from ReCRED.

These components are described in detail in the remainder of this section.

3.1.1.1 U-Prove engine implementation

The U-Prove engine implementation focuses primarily on the three entities that interact during the U-Prove protocol: the Prover (the user), the Issuer and the Verifier. The class diagram, presented below, includes also other entities which fulfill the whole protocol.

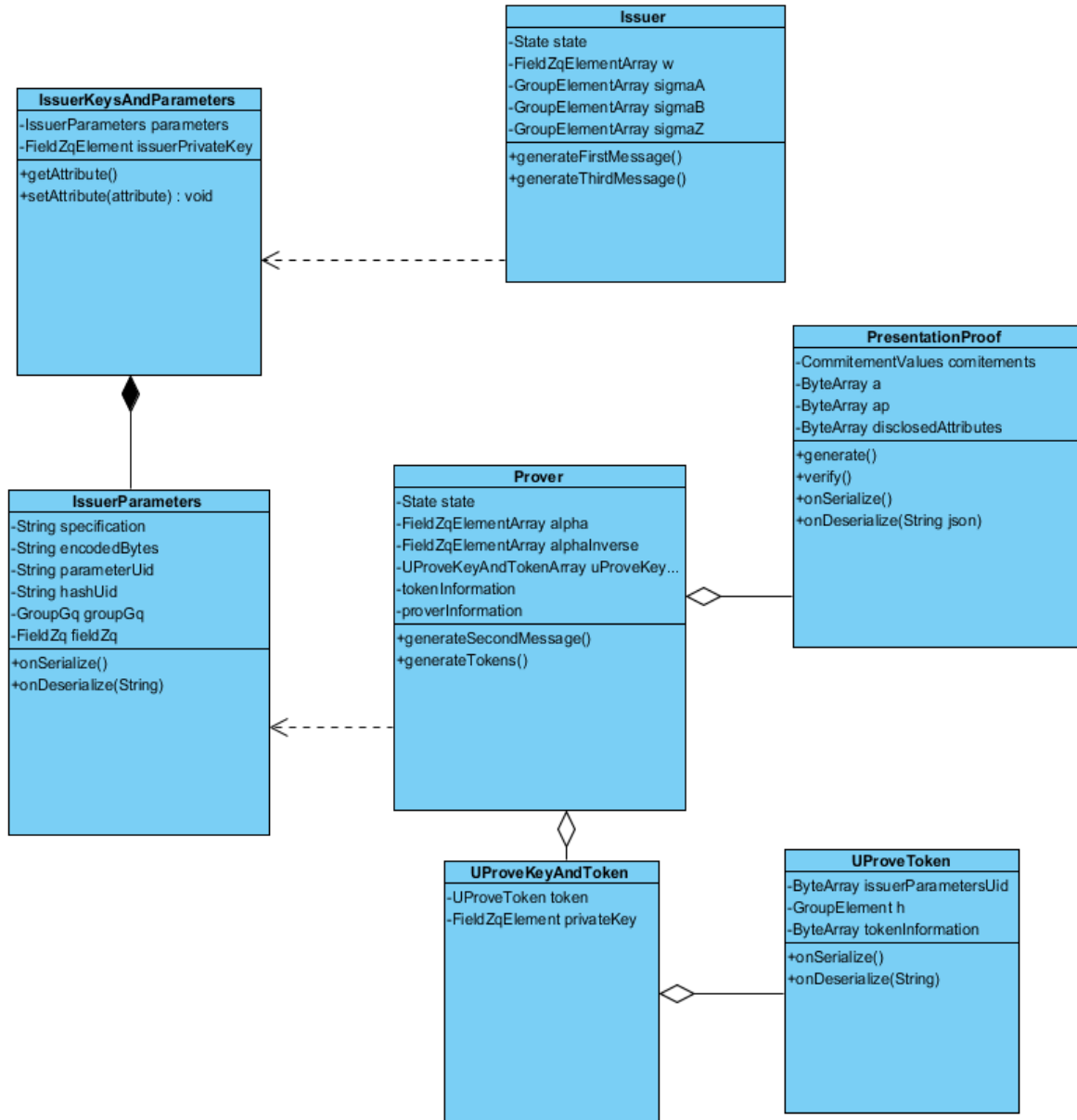


Figure 21 The class diagram for the U-Prove engine

3.1.2.1.1 U-Prove entities

1. **The Issuer entity**: Is the most complex entity from the U-Prove architecture and is responsible for generating the system parameters, the issuer parameters (based on the user request) and for creating two messages during the issuance protocol. In order to start the issuance protocol, the user sends to the issuer the specification of the token. In the generation of the issuer parameters

a number of variables have default values. These variable can be configured in order to customize the U-Prove algorithm. The table below shows which parameters should be provided by the user and which have the default values.

Parameter Name	Parameter Default Value	Mandatory
Number of Tokens	-	yes
Group Construction	Subgroup	no
IssuerParameterUid	-	yes
HashAlgorithm	-	yes
GroupName	SubgroupParameterSets.ParamSet_SG_2048256_V1Name	no
Token Information	-	yes
Prover Information	-	yes
Specification	null	no

Table 1 U-Prove parameters used during the issuance protocol

The code below illustrates the process of validating and generating the issuer parameters:

```

public IssuerKeyAndParameters generate(boolean supportDevice) throws
NoSuchAlgorithmException, InvalidUProveArtifactException {
    validate();
    GroupElement[] gValues = null;
    if (issuerParameters.getGq() == null) {
        if (this.parameterSet != null) {
            issuerParameters.setGq(this.parameterSet.getGroup());
            gValues = this.parameterSet.getG();
            if (supportDevice) {
                issuerParameters.setGd(this.parameterSet.getGd());
            }
            ip.UsesRecommendedParameters =
ParameterSet.ContainsParameterSet(this.ParameterSet.Name);
        }
        else
        {
            ParameterSet defaultParamSet =
IssuerSetupParameters.getDefaultParameterSet(this.groupConstruction);
            issuerParameters.setGq(defaultParamSet.group);
            if (useRecommendedParameterSet) {
                gValues = defaultParamSet.generatorsGroup;
                issuerParameters.setGd(defaultParamSet.getGd());
                issuerParameters.setUseRecommendedParameterSet(true);
            }
        }
    }
    FieldZqElement y0 = ProtocolHelper.generateIssuerParametersCryptoData(issuerParameters,
gValues, supportDevice);
    return new IssuerKeyAndParameters(issuerParameters,y0)
}

```

The three main methods from this entity are:

- *precomputation*(GroupElement gamma, FieldZqElement[] pregenaretdValues) - which performs the precomputation phase from the issuance protocol. The user may send to the user precomputed values for increasing the speed of the protocol;

- *generateFirstMessage()* - this method generates the first message object based on the information provided by the user in the previous phase. The engine also verifies that the issuance protocol is in the right phase before retrieving the message for the user;
- *generateThirdMessage(SecondIssuanceMessageComposite secondMessage)* - this is the third message which the user generates based on the given parameters. After this step, the issuer deletes the random elements generated during the previous phases.

This entity has a *State* parameter which maintains the state of the issuance protocol. If the Issuer gets a request to generate a U-Prove message, the state parameter is checked in order to validate the integrity of the issuance protocol.

2. **The Prover entity:** The functionality that this entity offers is similar to the Issuer. The main methods presented in this class are:

- *precomputation(GroupElement gamma,FieldZqElement[] pregenaretdValues)* - very similar to the function with the same name from the issuer side. It also precomputes some data to increase the speed of the issuance protocol;
- *generateSecondMessage(FirstIssuanceMessageComposite)* - this method generates the second message from the issuance protocol based on the first message received from the issuer;
- *generateTokens (ThirdIssuanceMessageComposite)* - generates an array of *UProveKeyAndToken* objects based on the third message received from issuer. The number of tokens to be generated is sent to the issuer by the user in a previous phase.

3. **The Verifier entity:** In our implementation the verification part is presented in the *PresentationProof* class. This class performs at first the generation of the proof on the Prover side, using the method *generate*. The presentation proof is then sent to the verifier which uses this proof in the verification process by calling the *verify* method. The generation of the proof, as mentioned in the protocol, may involve an additional device. The code below is a simplified version of the verify method which is the main method through which the integrity of the token is verified.

```
protected void verify(IssuerParameters issuerParameters,int[] disclosed,int[] committed,int
pseudonymAttribIndex,GroupElement gs, byte[] message, byte[] messageD,UProveToken uproveToken)
throws NoSuchAlgorithmException, NoSuchProviderException, IOException,
InvalidUProveArtifactException
{
    if (disclosed == null){
        disclosed = new int[] { };
    }
    Arrays.sort (disclosed);
    Group Gq = issuerParameters.getGq ();
    int n = issuerParameters.getEncodingBytes().length;
    boolean presentPseudonym = false;
    boolean verifyCommitments = (committed != null && committed.length > 0);
    if (verifyCommitments){
        Arrays.sort(committed);
    }
    ProtocolHelper.isTokenSignatureValid(issuerParameters, uproveToken);
    int dArraySize = disclosed.length + 2;
    GroupElement[] dBases = new GroupElement[dArraySize];
    FieldZqElement[] dExponents = new FieldZqElement[dArraySize];
    dBases[0] = issuerParameters.getG()[0]; dExponents[0] =
issuerParameters.getZq().getOne(); // g0^1
    dBases[1] = issuerParameters.getG()[n + 1];
    dExponents[1] = ProtocolHelper.computeXt(issuerParameters,
uproveToken.getTokenInformation(), uproveToken.isDeviceProtected()); // gt^xt
    FieldZqElement[] disclosedX = new FieldZqElement[disclosedAttributes.length];
```

```

        int aPreImageArraySize = 2 + (n - disclosed.length) +
        (uproveToken.isDeviceProtected() ? 1 : 0);
        GroupElement[] aPreImageBases = new GroupElement[aPreImageArraySize];
        FieldZqElement[] aPreImageExponents = new FieldZqElement[aPreImageArraySize];
        aPreImageBases[1] = uproveToken.getH(); aPreImageExponents[1] = this.r[0]; // h^r0
        int dIndex = 0; int uIndex = 1; int cIndex = 0; int pseudonymResponseIndex = 0;
        int[] commitmentResponseIndices = verifyCommitments ? new int[committed.length] :
null;
        for (int i = 1; i <= n; i++){
            if (UProveUtil.contains(disclosed,i){
                disclosedX[dIndex] = ProtocolHelper.computeXi(issuerParameters, i -
1,disclosedAttributes[dIndex]);
                dBases[dIndex + 2] = issuerParameters.getG()[i];
                dExponents[dIndex + 2] = disclosedX[dIndex];
                dIndex++;
            }
            else{
                aPreImageBases[uIndex + 1] = issuerParameters.getG()[i];
                aPreImageExponents[uIndex + 1] = this.r[uIndex]; // gi^ri
                if (presentPseudonym){
                    if (pseudonymAttribIndex == i) {
                        pseudonymResponseIndex = uIndex;
                    }
                }
                if (verifyCommitments){
                    if (UProveUtil.contains(committed,i)){
                        commitmentResponseIndices[cIndex] = uIndex;
                        cIndex++;
                    }
                }
                uIndex++;
            }
        }
        GenChallengeReturnType genChallengeReturnType =
ProtocolHelper.genChallenge(issuerParameters, uproveToken, this.a, pseudonymAttribIndex,
this.ap, this.ps, message, messageD, disclosed, disclosedX,
committed, this.commitments);
        FieldZqElement c = genChallengeReturnType.challenge;
        aPreImageBases[0] = Gq.multiExponentiate(dBases, dExponents);
        aPreImageExponents[0] = c.negate(); // g0.gt^xt.Product[gi^xi]_(for disclosed i)
        HashFunction hash = issuerParameters.getHashFunction();
        hash.update(Gq.multiExponentiate(aPreImageBases, aPreImageExponents));
        byte[] test = hash.getByteDigest();
        if (!UProveUtil.sequenceEqual(this.a,test)){
            throw new InvalidUProveArtifactException("Invalid presentation proof");
        }
    }
}

```

3.1.2.1.2 U-Prove protocol flow

During the protocol, it is assumed that the attributes are known to both the Prover and the Issuer and the latter should not verify the validity of the data. In the diagram below the flow of the application during the issuance protocol is described taking into account the previous considerations. The verification process is very straight-forward and thus is not described here through a sequence diagram.

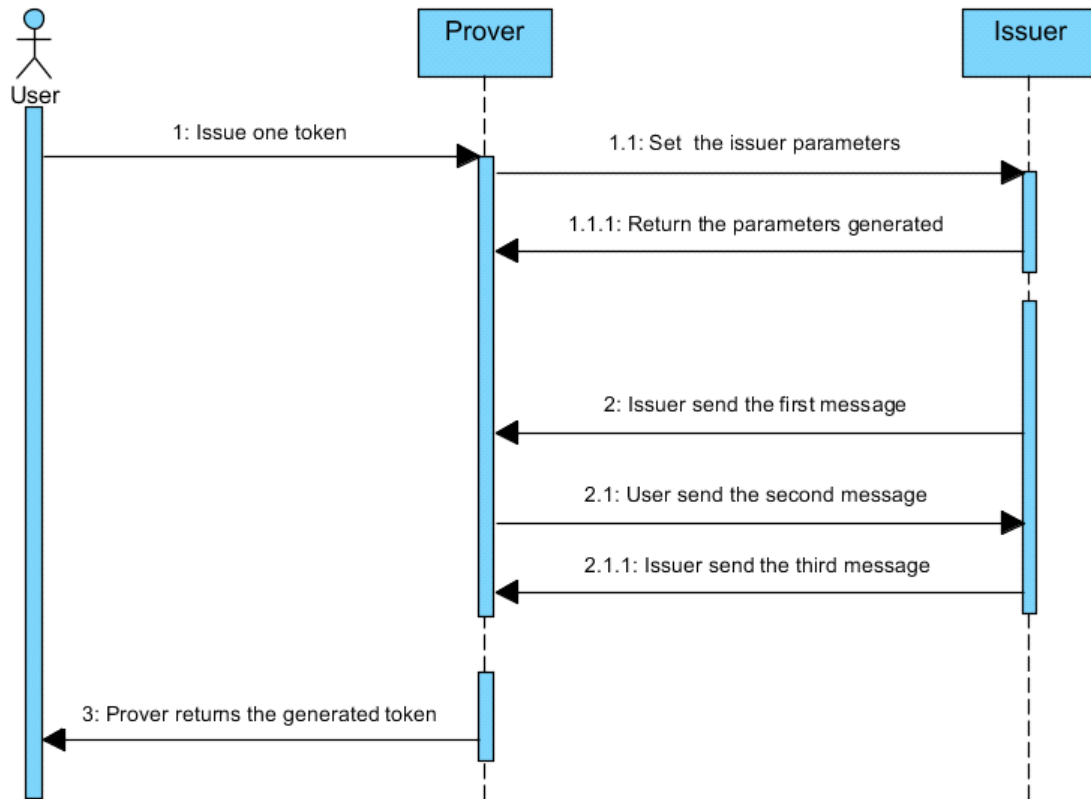


Figure 22 The issuance protocol from U-Prove

3.1.2.1.3 U-Prove additional considerations

In order to test the whole protocol, we used the official test vectors from Microsoft and created unit tests which validate the correctness of the implementation.

As mentioned before, the implementation supports all the specifications provided by the Microsoft documentation. Some features provided by this implementation are:

- the U-Prove token may be protected by an additional device;
- a wide range of security level values (from 80 to 256) are allowed;
- both the ECC and Subgroup constructions are supported.

3.1.2.2 U-Prove Web Server

In order to be integrated into the ReCRED architecture, over the engine's implementation of U-Prove we have developed a service API which accepts and produces JSON data. This web service includes two main subservices: *UProveIssuerService* and *UproveVerifierService*.

In order to create the U-Prove REST API we use the Java programming language and the Jersey library for building the web server. In order to manage the dependencies more properly and deploy the war archive more easily, we use Maven.

3.1.1.1.1 U-Prove Issuer Service

The main aim of this subservice is to implement the issuance process, which provides the U-Prove token. The most important methods which the service offers are listed in the table below.

Name	Method	Description	Error Codes
<i>handshake</i>	GET	Used only once at the beginning of the issuance protocol. This method produces a uniquely random string which will be used for requesting other U-Prove messages.	<i>MAX_ACTIVE_SESSIONS_MESSAGE</i> - if the maximum number of concurrent sessions has been reached; <i>INVALID_SESSION_EXCEPTION_MESSAGE</i> – if the process of adding a new session failed. This thing may happen if the newly generated value for the session already exists in the memory of the service.
<i>createIssuerSetupParameters</i>	POST	Generates a new fresh set of issuer parameters based on the given specification provided by the user. The specification must be serialized in JSON format and the request contains also the sessions id previously generated.	<i>INVALID_SESSION_EXCEPTION_MESSAGE</i> – if the sessionId value was not found in memory or expired since was generated; <i>JSON_SERIALIZATION_EXCEPTION_MESSAGE</i> – if some error occurs during the serialization process; <i>HASH_EXCEPTION_MESSAGE</i> – if an error occurs during the hash operations. These errors may occur if the specified hash algorithm was not found or if the <i>PROTOCOL_EXCEPTION_MESSAGE</i> – if other U-Prove exception occurs during the method call.
<i>generateFirstMessage</i>	GET	Performs the first step of the issuer during the issuance protocol. The user must provide additional details in order to start the issuance protocol (number of tokens to generate, token information field value etc.). The session id value is required in this step also.	<i>INVALID_SESSION_EXCEPTION_MESSAGE</i> – if the sessionId value was not found in memory or expired since was generated; <i>JSON_SERIALIZATION_EXCEPTION_MESSAGE</i> – if some error occurs during the serialization process; <i>PROTOCOL_EXCEPTION_MESSAGE</i> – if other U-Prove exception occurs during the method call. The most frequent error that maps to the exception message is the invalid state error raised by the engine during the <i>getFirstMessage</i> method from the U-Prove engine.
<i>generateThirdMessage</i>	POST	Generates the third message of the U-Prove protocol. The input of this method is a serialized version of the second U-Prove message generated by the prover.	<i>INVALID_SESSION_EXCEPTION_MESSAGE</i> – if the sessionId value was not found in memory or expired since was generated; <i>JSON_SERIALIZATION_EXCEPTION_MESSAGE</i> – if some error occurs during the serialization process;

	<p><i>PROTOCOL_EXCEPTION_MESSAGE</i> – if other U-Prove exception occurs during the method call. The most frequent error that maps to the exception message is the invalid state error raised by the engine during the <i>getFirstMessage</i> method from the U-Prove engine.</p>
--	---

Table 2 – U-Prove REST API

3.1.1.1.2 U-Prove Verifier Service

Contains only one method which performs the verification of the U-Prove token. This method is described below.

- *verifyToken* – Performs the verification step from the U-Prove protocol. The user must send to the server the presentation proof as well as the token and the attribute he wants to disclose. Beside the disclosed parameters, the user may send to the verifier the committed attributes, if any. This step doesn’t require the session id parameter. It may return similar error codes in case of failure. If the verification process succeeded, the *Success* message is returned.

3.1.3 Trusted Execution Environment

GlobalPlatform’s Trusted Execution Environment (TEE) offers safe execution of authorized security software, known as “Trusted Applications”, enabling it to provide end-to-end security by enforcing protected execution of authenticated code, confidentiality, authenticity, privacy, system integrity and data access rights. Hence, TEE in the context of ReCRED will provide a secure platform for storing and handling sensitive user information on her mobile device. TEE defines a distinction between Normal World, where common OS and applications are executed, and Secure World, which hosts Trusted OS and applications. In ReCRED, only a subset of the user’s operations will be executed inside Secure World (such as storing secret keys in TEE Trusted Storage), while the rest of the operations will remain in the Normal World. The communication between Normal World and Secure World will be achieved by utilizing the TEE Client API, defined in GlobalPlatform specification.

Idemix is a protocol which will be implemented in ReCRED, that allows user authentication without divulging any personal data. In Idemix, the User identity stores sensitive user information which typically requires interaction with the Issuer identity. This interaction (also known as Issue protocol), in its simplest form, involves two rounds: (a) the User submits a request containing her attributes and (b) the Issuer certifies the fact that the User has the claimed attributes by returning a credential. More generally, the credentials can be generated through a multi-round interaction between the two identities. These credentials can be later used by the user to convince a Verifier identity that she has a certain set of attributes. Critical operations of the User identity should benefit by the leveraged security provided in TEE, thus offering a greater level of security to the end user.

Each User in Idemix, is required to generate a random secret key, which should not be made publicly available. The key should be stored safely in user’s device for later use. Moreover, the credentials issued to a User, should also be safely stored in her device, to be used during the verification process. To this end, TEE’s Trusted Storage will provide to ReCRED a safe storage place to store User’s secret key and issued credentials.

User's secret key constitutes crucial information in terms of security, and should not be made available to Normal World applications. Thus, cryptographic operations that require knowledge of the User's secret key, cannot be executed in Normal World. Trusted Applications designed to run inside TEE will be implemented in ReCRED, to calculate specific values (such as U and $sHat$) required during the execution of the Issue Protocol. An example of the Idemix issuing protocol workflow implemented inside the TEE by means of OpenTEE is depicted in Figure 23.

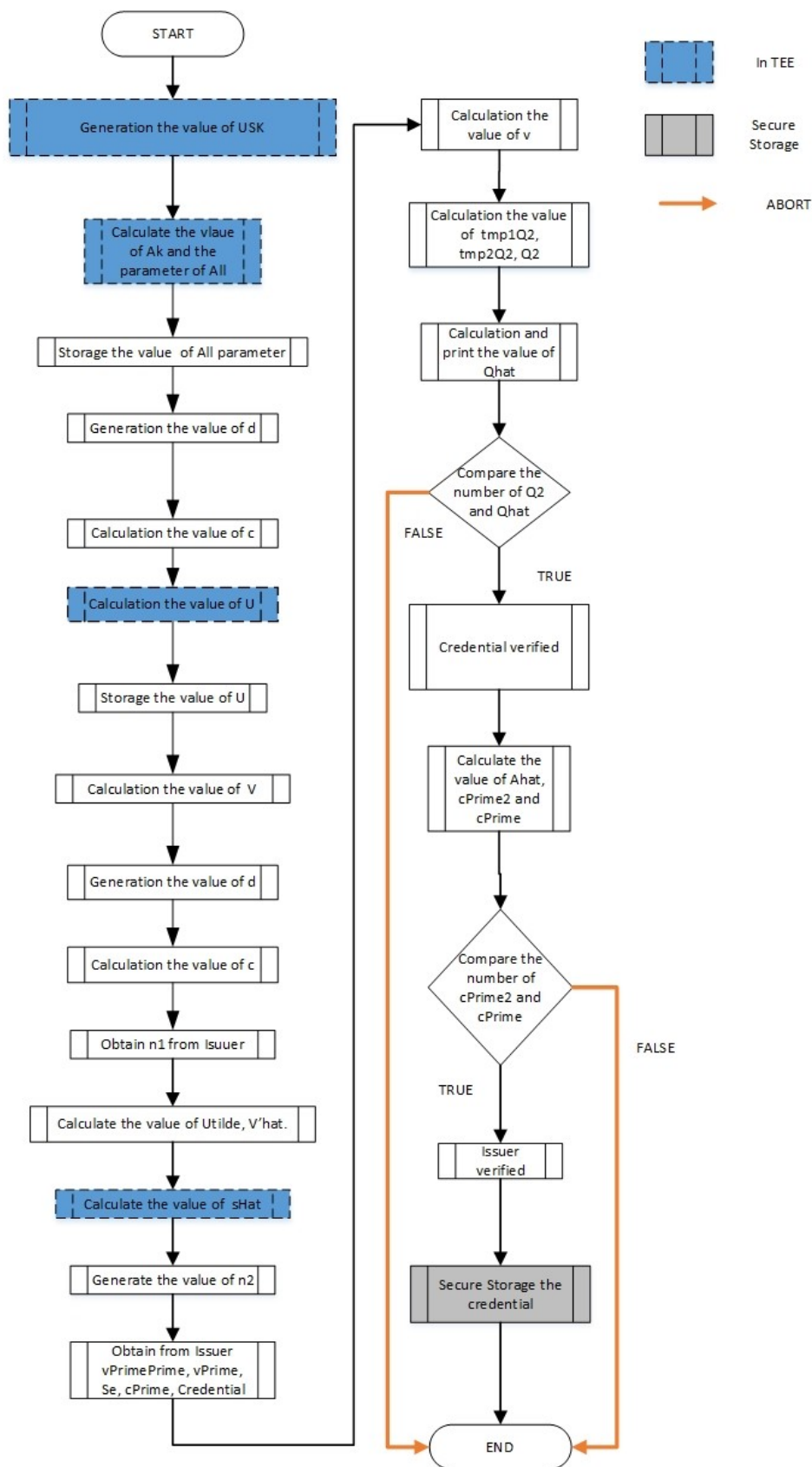


Figure 23: Idemix Issue protocol workflow implemented in TEE by means of OpenTEE

3.1.4 Consent Management

3.1.4.1 Consent Management Back-end

The consent management backend in Figure 24 is a API enabled module that contains standard functionalities for creating policies & consent. The main components are:

- **Consent management API:** the consent management API allows for the administration of policies. Users and identity providers can create, read, update and remove policies using this component. The consent management API is able to handle all specifics of consent management.
- **The policy enforcement engine** basically allows clients to use on functionality: to get a decision whether information can or cannot be disclosed.
- A centralized **consent policy data store** is used to store all types of policies
- **An authorization layer** is put in place on top of all APIs. Each of the endpoints of the consent management module is protected using OAuth2. As all policies are stored inside one database, it must be made sure that users and identity providers can only manage their own policies. To do this, for each request, the authorization layer will verify whether an access token is available. Using this access token, the authorization layer is able to fetch the identity of the user. The authorization layer then inspects the contents of the request to make sure the rule belongs to the particular person or whether it is allowed the specific decision is requested.

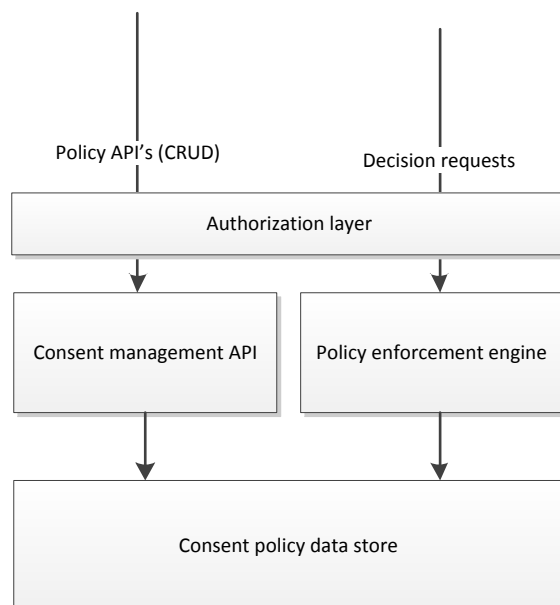


Figure 24 Consent Management Back-end architecture

3.1.4.2 Consent Management Mobile Application

The Consent Management mobile front-end has been implemented as an Android mobile application that allows the users to define their consent for their various identity attributes by defining policies regarding the Identity Providers and Service Providers to which their attributes should be revealed. These consent policies are then enforced, every time the user wishes to prove an identity attribute to a verifier.

The mobile app communicates with the Consent Management back-end, allowing the end-users to access the following functionality:

- Create new consent policies, by hiding specific identity attributes (or groups of attributes) from Identity Providers and/or Service Providers
- View the consent policies that he has created, and group them by attribute, Identity Provider or Service Provider
- Modify or delete consent policies that he has already created

3.1.4.2.1 Create new Consent Policy

The user can create new consent policies through a two-step procedure: the user defines the attribute(s) that he wants to hide and then he defines the Identity Provider(s) or Service Provider(s) from whom he wants to hide these specific attributes (Figure 25). Hiding an attribute from an Identity Provider means that this attribute cannot be transferred to that Identity Provider (or revealed to it, in cases where the Identity Provider might act as a Service Provider). Hiding an attribute from a Service Provider means that this attribute cannot be transferred to that Service Provider.

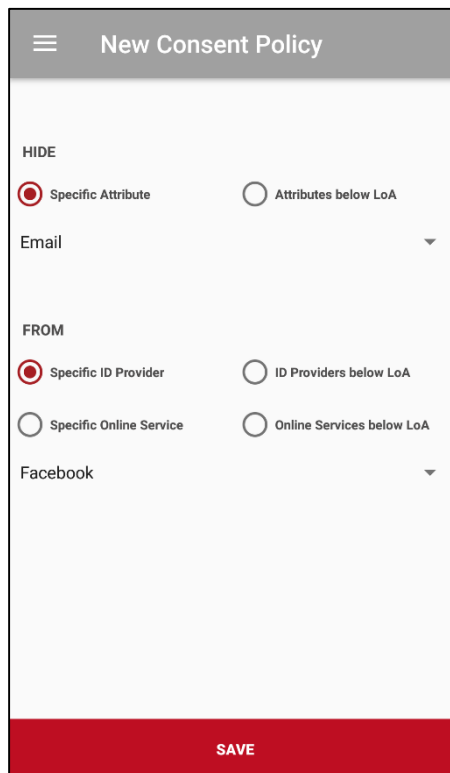


Figure 25 Create new consent policy page

More specifically, at first the user must select the attributes that he wants to hide. There two options here:

1. The user can select to hide a specific identity attribute, which he can select from a drop-down list (Figure 26).
2. The user can select to hide all his identity attributes that are below a certain Level of Assurance (LoA), which he can also select from a drop-down list (Figure 27)

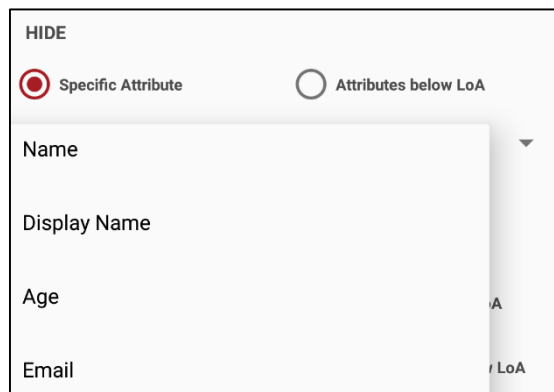


Figure 26 Selection of specific identity attribute

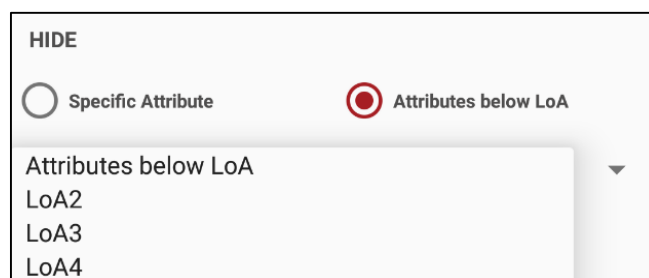


Figure 27. Selection of identity attributes below a certain LoA

After that, the user must select the Identity Providers or Service Providers from whom the selected attribute(s) will be hidden. Here, there are four possible options:

1. The user can select a specific Identity Provider (Figure 28)
2. The user can select all the Identity Providers below a certain LoA (Figure 29)
3. The user can select a specific Service Provider (Figure 30)
4. The user can select all the Service Providers below a certain LoA (Figure 31)

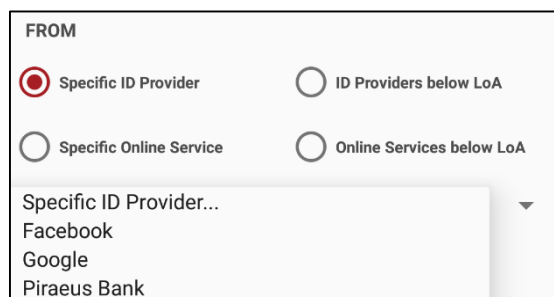


Figure 28. Selection of specific Identity Provider

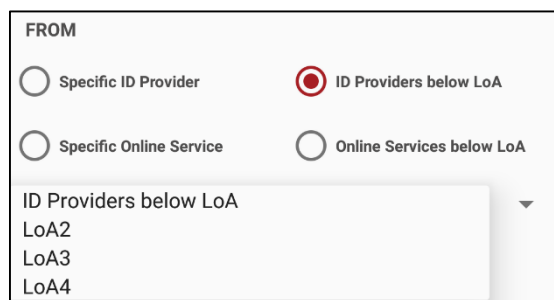


Figure 29. Selection of Identity Providers below a certain LoA

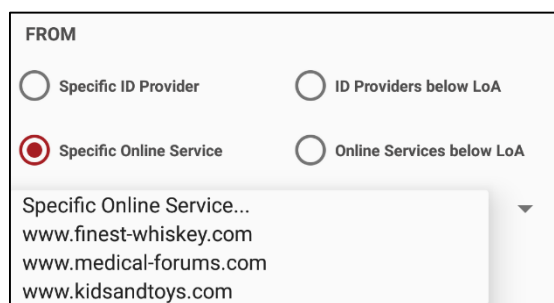


Figure 30. Selection of specific Service Provider

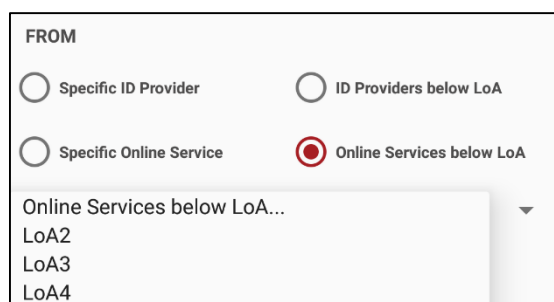


Figure 31. Selection of Service Providers below a certain LoA

3.1.4.2.2 View created consent policies

The user can see a list with all the consent policies that she has created. These attributes are grouped under three different sections:

3.1.4.2.2.1 View created consent policies by an Identity attribute

Here, the user can see a list with all his identity attributes and how many consent policies he has created for each attribute (Figure 32). After selecting a specific attribute, the user can see all the consent policies that he has created for it (Figure 33), either explicitly for that attribute or for an LoA where the attribute is included.

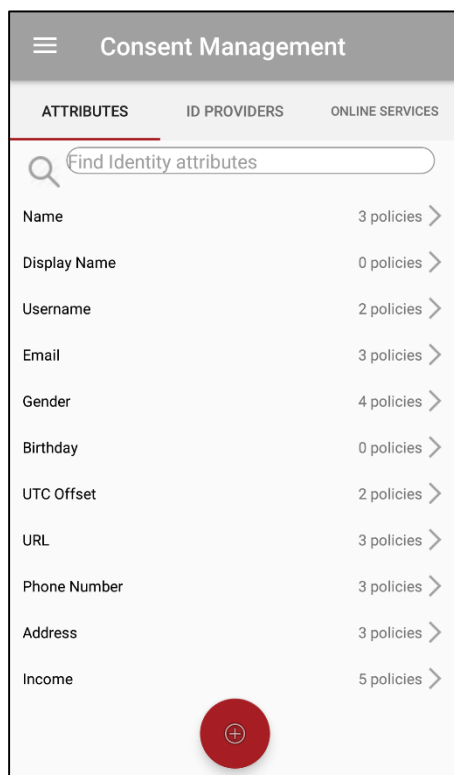


Figure 32. List of a user's identity attributes

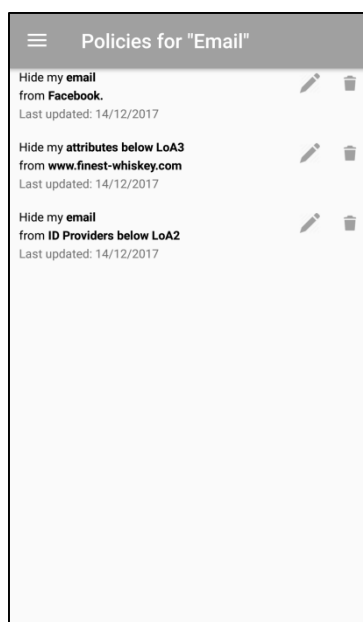


Figure 33. View created policies per identity attribute

3.1.4.2.2.2 View created consent policies by an Identity Provider

Here, the user can see a list with all his Identity Providers and how many consent policies he has created regarding that Identity Provider (Figure 34). After selecting a specific Identity Provider, the user can see all the consent policies that he has created in order to hide attributes from it (Figure 35), either explicitly for that Identity Provider or for an LoA where the Identity Provider is included.

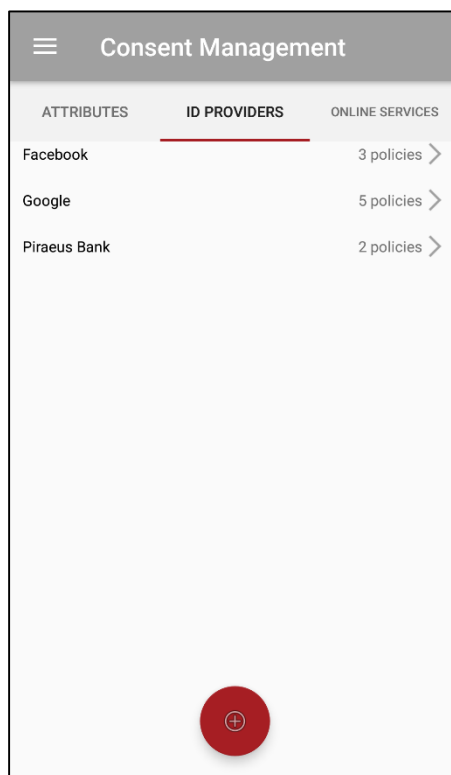


Figure 34. List of Identity Providers

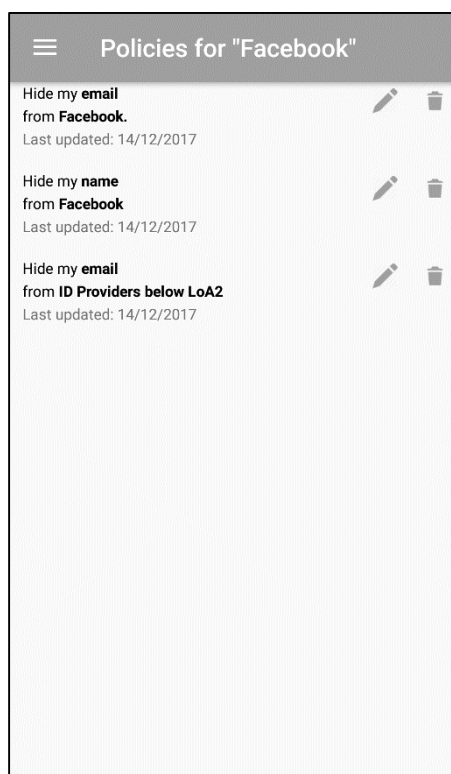


Figure 35. List of created policies per Identity Provider

3.1.4.2.2.3 View created consent policies by Service Provider

Here, the user can see a list with all his Service Providers and how many consent policies he has created regarding that Identity Provider (Figure 36). After selecting a specific Service Provider, the user can

see all the consent policies that he has created in order to hide attributes from it (Figure 37), either explicitly for that Service Provider or for an LoA where the Service Provider is included. The user can also modify or delete any policy.

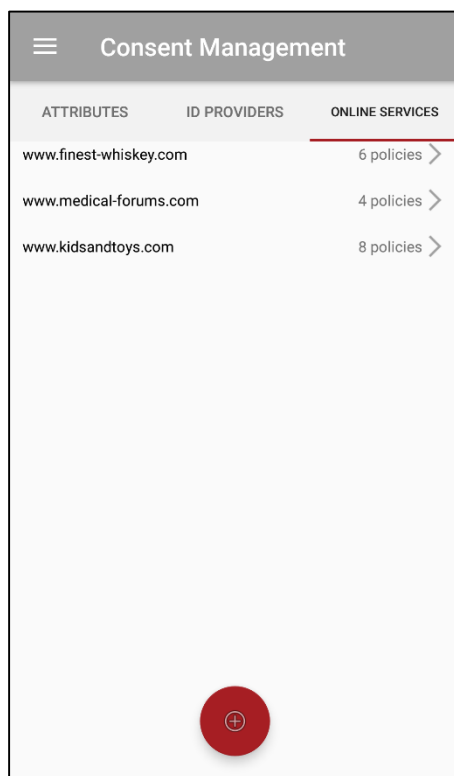


Figure 36. List of Service Providers

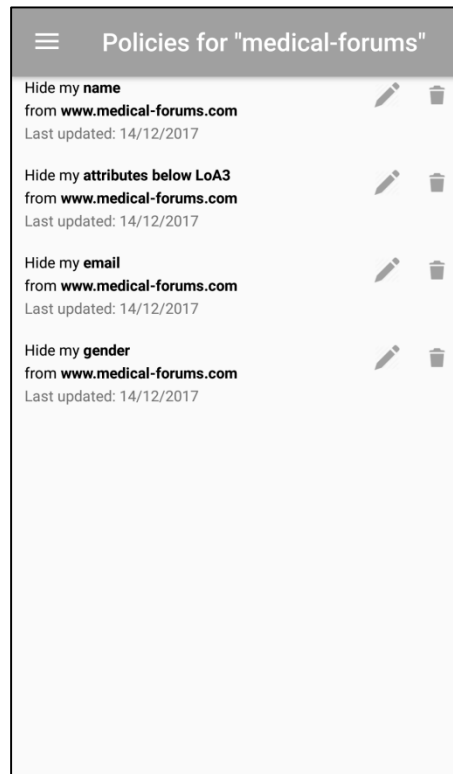




Figure 37. List of created consent policies per Service Provider

3.1.4.2.3 Manage created consent policies

At any point while viewing her consent policies, the user has the option to modify them or even delete them.

In order to modify a policy, the user can tap on the  icon beside it, in which case the *Edit Consent Policy* screen is appeared, filled-in with the details of the selected policy. The user can modify the policy as he wishes and save her changes.

In order to delete a policy, the user can tap on the  icon beside it, in which case a confirmation dialog is shown and the consent policy is deleted.

3.1.4.3 Consent Management Web Front-end

The functionality of the consent management web front-end is the same as the functionality of the mobile application.

As depicted in Figure 38, the Web frontend architecture consists of two parts:

- The interface itself will be based on AngularJS to have a smooth user experience. Bootstrap UI is used for layouts.
- A backend server component will be put in place to do the heavy lifting and sending the correct requests to the consent management API.

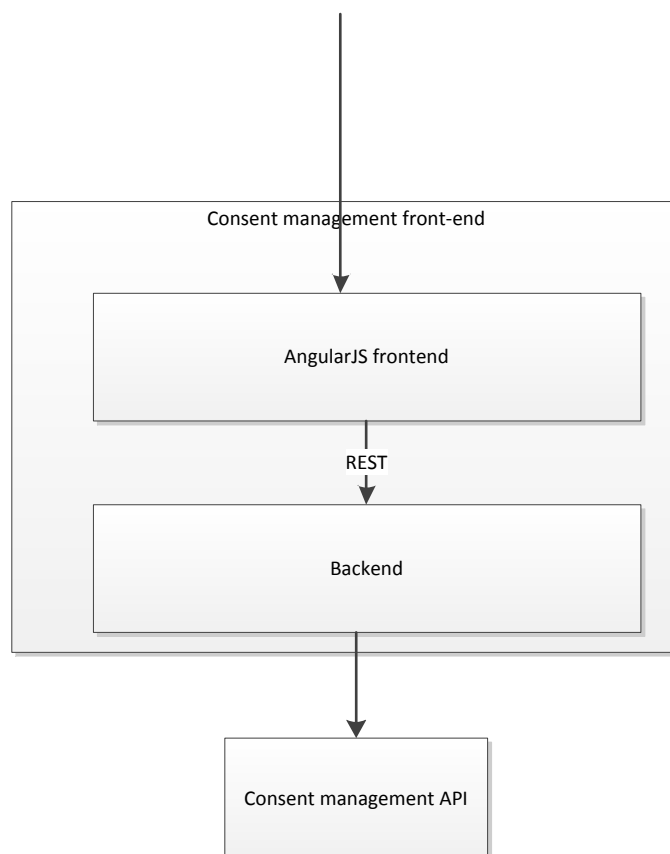


Figure 38 Consent Management Web Front-end architecture

3.1.5 De-anonymization Risk Assessment

Risk management has been integrated in Identity Profile Management in order to offer a friendlier and more coherent User Experience. De-Anonymization Risk is calculated for ID Providers Data, Financial Information, and Service Providers.

The de-anonymization risks that are taken into account involve risk of attribute value inference, and risk of user identification also known as k-Anonymity.

The risk of user identification is a metric of how many users exist with the same (known) attribute values, therefore the risk corresponds to the probability that a user is uniquely identified based on the revealed attribute values. In statistical terms, it is an indication of where the user fits within the user population as this is segmented based on revealed attribute values.

Attribute value inference is essentially the risk that an ID Provider guesses the value of an unknown user attribute based on the known attribute values of this user. In statistical terms, it is an indication of where the user fits within the user population as this is segmented based on the revealed attribute values and the unrevealed attribute that the risk is calculated for.

Please refer to the corresponding chapter for a more detailed description of risks related with Service Providers.

3.1.5.1 ID Provider Fields Risk

ID	Provider	Name	k-Anonymity	Display Name	Preferred Username	Gender	Birthday	Utc Offset	Email	Verified Email	Uri
1079	Facebook		1/3			male		1		✗	
1087	Facebook		1/3			male		2		✗	
1095	Facebook		3/3			male		1 50%		✗	

Showing 1 - 3 of 3 items.

Screen 1: ID Provider Field De-Anonymization Risks

k-Anonymity is displayed as “{Number of users with similar revealed attributes} / {Total number of users}”.

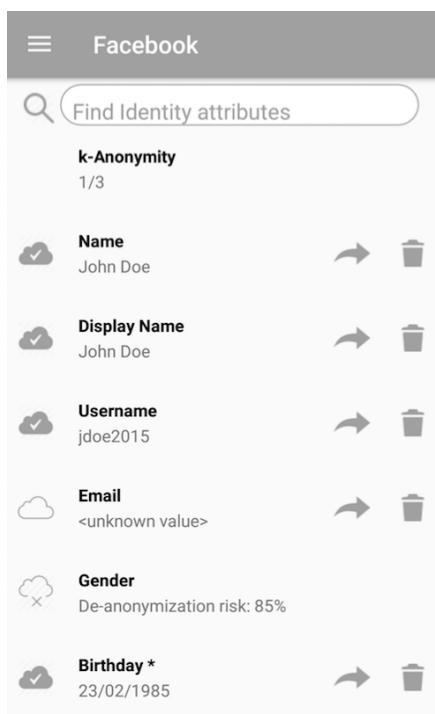
Each of the attributes where de-anonymization is applicable is associated with a percentage that represents the possibility that the Identity Provider may correctly guess the value of the attribute.

Provider	In ID Provider	Risk
Name	✓	
k-Anonymity		3/3
Display Name	✓	
Preferred Username		
Gender	male ✓	*
Birthday		
Utc Offset	1	50%
Email		
Verified Email	✗	
Uri		
Phone Number		
Photo		

Screen 2: ID Provider Field De-Anonymization Risks - Detail

A detailed view of the ID Provider Fields Risks displays the de-anonymization risks next to applicable attributes. A star “*” symbol is used to indicate which attributes have been used for risk calculation, i.e. the *known* attribute values.

In a similar way, the Android app displays the k-anonymity rating for any given Identity Provider. It also displays the de-anonymization risk for identity attributes that are not known to the Identity Provider. A star “*” symbol is used to indicate which attributes have been used for risk calculation.



Screen 3: ID Provider De-Anonymization Risks (Mobile)

Calculation is based on the assumption that the Id Provider in question has a dataset of user records similar to the one in Id Repository. Actual risk may be higher or lower than the calculated depending on whether the Id Provider’s dataset is more or less accurate than the Id Repository one.

3.1.5.2 Financial Information Risk

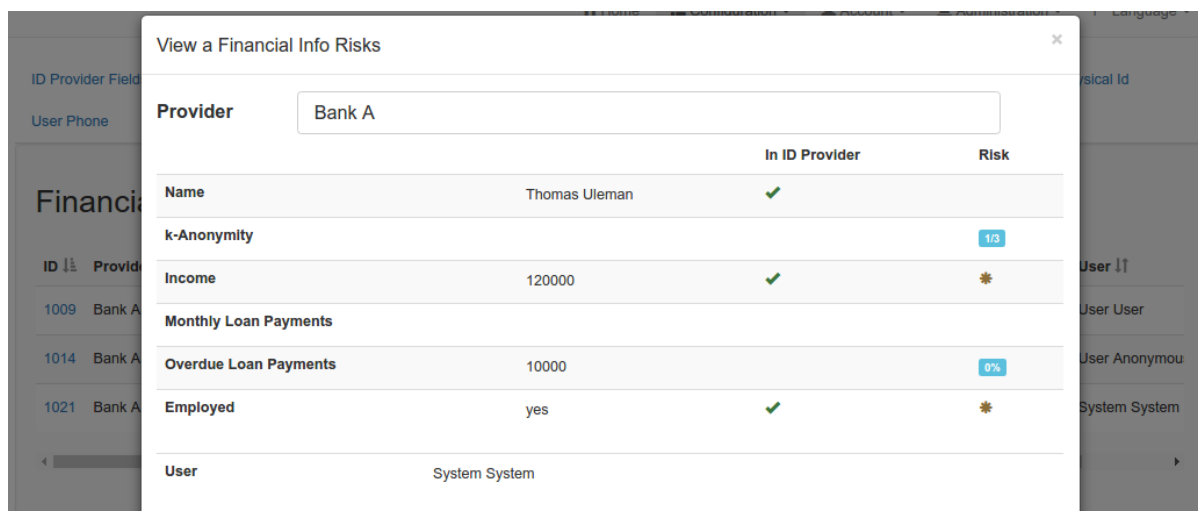
De-anonymization risk for Financial Information is calculated following the same logic.

Financial Info Risks								
ID	Provider	Name	k-Anonymity	Income	Monthly Loan Payments	Overdue Loan Payments	Employed	User
1009	Bank A	James Thorn	1/3	123000			✓	User User
1014	Bank A	Andrea Wegeman	1/3	122000 0%		12000 0%	✓	User Anonymou:
1021	Bank A	Thomas Uleman	1/3	120000		10000 0%	✓	System System

Showing 1 - 3 of 3 items.

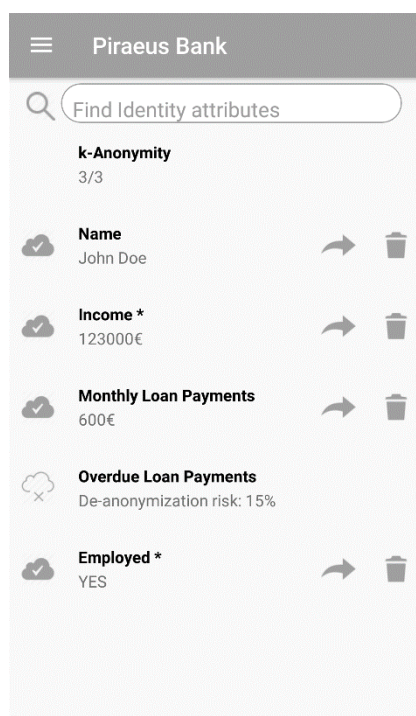
Screen 4: Financial Information De-Anonymization Risks

And the same layout is used for the detailed view of Financial Information records.



Screen 5: Financial Information De-Anonymization Risks - Detail

The Android app displays the Financial Information Risk in a similar way.



Screen 6: Financial Information De-Anonymization Risks (Mobile)

3.1.5.3 Service Providers

The risks associated with Service Providers are separated in two different categories depending on whether they use P-ABAC or not to authenticate their users.

Without P-ABAC authentication

If the user uses plain Open ID Connect to access a Service Provider, then anonymity is forfeited by definition. The Service Provider will be able to uniquely identify the user because the user will explicitly allow the Service Provider to access the user’s identity attributes.

Therefore, the only risk of interest is the risk of Attribute Value Inference, regarding profile attributes that the user forbids access to. The risk model is the same as described in earlier chapters about Attribute Value Inference and the calculations are performed on the ID Repository dataset, based on the assumption that the Service Provider has a statistically similar dataset. Actual risk may be higher or lower depending on whether the Service Provider has a more or less accurate dataset than the Id Repository one.

With P-ABAC authentication

Idemix and U-Prove are protocols that offer anonymity as well as un-traceability. This means that a Service Provider cannot in principle identify a user, and also cannot conclude whether different service sessions involve the same user.

However, and only within the context of an ongoing session, a Service Provider may be able to match the user against a population of users with similar attributes. Also, the Service Provider may attempt to infer the value of an unrevealed attribute, after having matched the user against a population of users with attributes similar to the ones revealed for the ongoing session.

Note, that if the Service Provider collects population data based solely on Idemix or U-Prove sessions, this data may not represent a statistically correct population model because sessions are un-linkable with one another. Consider a scenario where a user attempts 99 sessions with the same attributes while another user attempts a single session with a different set of attribute values. In such a scenario, the collected dataset includes 99 similar records and a single different one, which however correspond to only two users. Unless the population model is known, no meaningful risk can be calculated.

Therefore, it is assumed that the Service Provider has obtained in some other way a dataset with proper population distribution against which users are matched and attributes may be inferred. The risk calculated by ReCRED is based on the assumption that the Service Provider has obtained a dataset similar to the dataset in Id Repository. The actual risk is higher or lower depending on whether the Service Provider’s dataset is more or less accurate compared to the dataset of Id Repository.

3.1.6 P-ABAC and FIDO Integration

Privacy-Preserving Attribute-Based Access Control (P-ABAC) is emerging as a means for reliably authenticating users to services while preserving their privacy. Idemix and U-Prove are among the most well-known mechanisms, and are being integrated in the ReCRED architecture components mainly through the activities of WP5, WP4 and WP3.

One way of taking advantage of FIDO in the ABAC architecture is employing the FIDO protocols in the authentication phase between the User Device and the Issuer. Indeed, in this phase, the issuer needs to know the identity of the user in order to disclose a cryptographic credential with her attributes. Thus, using FIDO would allow for a reliable and password-less P-ABAC credential issuance. However, this would be far from bringing to the User-centric mobile device authentication world the advantages of P-ABAC.

A tighter integration of P-ABAC mechanisms in FIDO allows instead users to authenticate through their devices to Online Services while preserving their privacy, while online services can attract more users to their platforms.

We here propose an integration of P-ABAC in the FIDO UAF protocol. An architectural overview is provided in Figure 39.

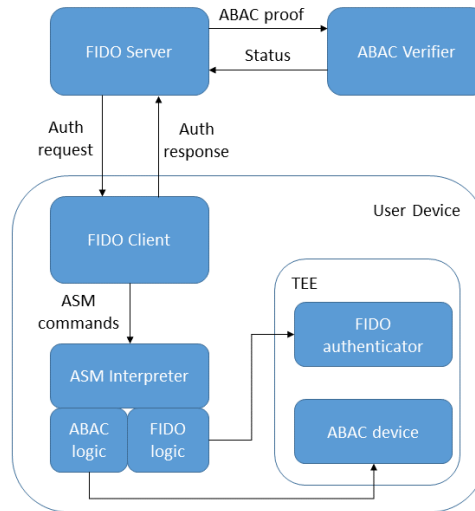


Figure 39 P-ABAC-FIDO Proposed Integrated Architecture

Figure 40 summarizes the FIDO UAF authentication process as defined by the FIDO specification [38]. Please note that the “RP Web App and Web Server” component referenced in the FIDO specification corresponds to the *Identity Provider* in the ReCRED architecture, and thus the following diagrams have been adapted to reflect this mapping.

For the integration that we are proposing, we assume that the P-ABAC credential issuance phase, which is not described here, has taken place before the authentication phase. Moreover, we assume that using a well-established special constant value for the username and public key triggers the P-ABAC-FIDO mechanism. This allows for the coexistence of the “normal” FIDO authentication mechanism along with the P-ABAC-FIDO “credential show” mechanism.

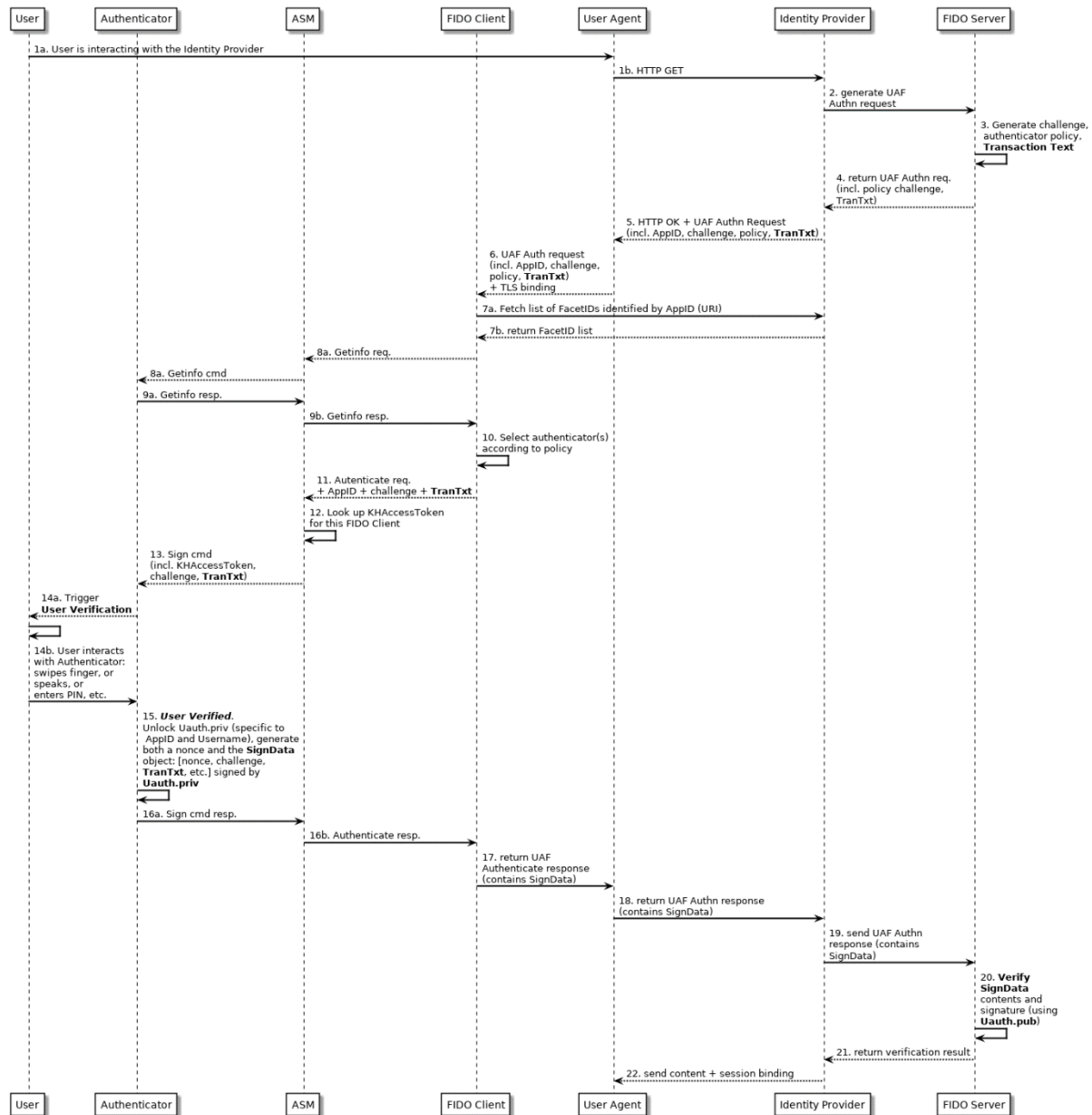


Figure 40 Authentication process from the FIDO UAF specification

Starting from the standard FIDO authentication mechanism depicted in the figure, we substitute the public-key based identification with a privacy-preserving attribute-based authentication: instead of the public-key cryptographic operations in steps 15 and 20, we employ a privacy-preserving attribute proving. Furthermore, the FIDO policy in steps 3, 4, 5, 6, 10 is replaced with a P-ABAC policy.

The modified protocol is depicted in Figure 41 below.

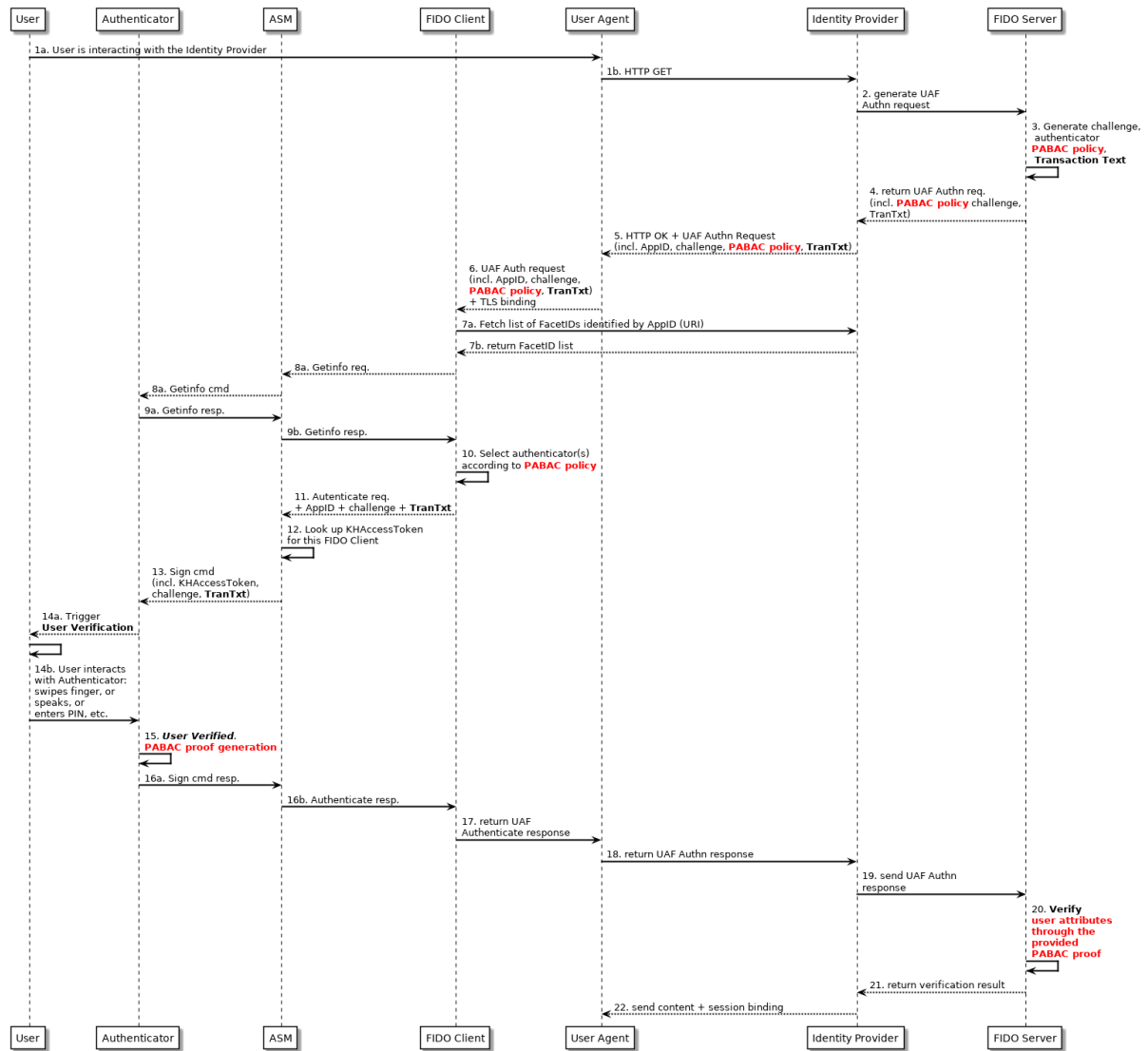


Figure 41 P-ABAC-FIDO integrated authentication protocol - proposed changes to the FIDO UAF specification are highlighted in red

When the user tries to access an unauthorized Web resource, the FIDO server replies with an `AuthenticationRequest` message as described in [38]. The FIDO server transmits to the client the required P-ABAC attributes through a FIDO extension as defined in the following data structures.

```
Dictionary AuthenticationRequest {
    required OperationHeader header;
    required ServerChallenge challenge;
    Transaction[ ] transaction;
    required Policy policy;
}
Dictionary Policy {
    required MatchCriteria[ ][ ] accepted;
    MatchCriteria disallowed;
}
Dictionary Extension {
    required DOMString id; /* Bind to 'P-ABAC attribute' */
    required DOMString data; /* Required attribute encoded as base64 */
    required boolean fail_if_unknown; /* Bind to true */
}
```

After the required attributes reach the software authenticator module through FIDO ASM, the P-ABAC proof is generated and sent back to the server serialized in a FIDO extension which is encapsulated in the FIDO `AuthenticatorSignAssertion` structure.

```
Dictionary AuthenticatorSignAssertion {
    required DOMString assertionScheme;
    required DOMString assertion;
    Extension[ ] exts;           /*Serialized P-ABAC proof*/
}
```

The latter structure is embedded in the FIDO `AuthenticationResponse` dictionary which is processed by the server.

3.1.7 Credential Backup

According to the ReCRED architecture, Identity Providers may issue credentials that ReCRED users store in their mobile devices. These credentials are stored in the mobile device and are encrypted using the TEE, if it exists, in the mobile device.

ReCRED offers Credentials Backup & Restore functionality as a failover mechanism in case the user's device is lost, stolen or broken and credentials must be restored to a new device.

In order to facilitate the Backup & Restore functionality, ReCRED includes three components developed specifically for this purpose:

- A backend that implements all the necessary business logic and performs the actual data transactions with the Identity Repository.
- A web application that offers users an overview of the credentials that have been backed-up in the Identity Repository.
- A mobile application that allows users to back-up credentials to the Identity Repository, as well as restore credentials to the mobile device.

3.1.7.1 Backend

The backend is a Java application that provides a REST API offering the following methods:

- Create Credential (POST `/api/credentials`): this is the method that accepts a new credential and saves it in Identity Repository. Essentially, this is the back-up method.
- Update Credential (PUT `/api/credentials`): overwrites an existing credential with a new version. It is a refresh of the backed-up credential.
- Get All Credentials (GET `/api/credentials`): will return all credentials if the user has administrator privileges, otherwise only the credentials of the current user.
- Get Credential (GET `/api/credentials/:id`): will return the credential with the specified id. The id is a storage specific identifier which is unique for every credential. The id is one of the properties returned for each credential by “*Get All Credentials*”.
- Delete Credential (DELETE `/api/credentials/:id`): will delete the credential with the specified “*id*”.
- Search Credentials (SEARCH `/api/_search/credentials?query=:query`): returns the results of a search operation performed on credentials based on parameter “*:query*” where query is a sentence containing text to look for in attribute properties.

The backend is accessed both by the web application and the mobile application that offer front-end interfaces to the backend API.

3.1.7.2 Mobile Application

The following functionalities are currently offered:

- Display all credentials currently available at the mobile device.
- All credentials available in the device can be encrypted by TEE if it exists.
- Credentials can be restored from the Identity Consolidator server using the 3rd Party API.
- The credentials in the device can be backed up in the Identity Consolidator server.

In order to facilitate the Backup & Restore functionality, the Identity Consolidator, includes backend functionality that offers the corresponding backup and restore API methods. Also, a web frontend

3.1.7.2.1 Main menu

In the main menu of the Credentials Backup & Restore application a user can see all the local cryptographic credentials stored in the device as well as all the remote credentials of the user that are available for backup.

Yet the way for authenticating the user of the device to the remote server for having access to the backup data has to be resolved. Currently the remote backup service is OAuth v0.2 secured.

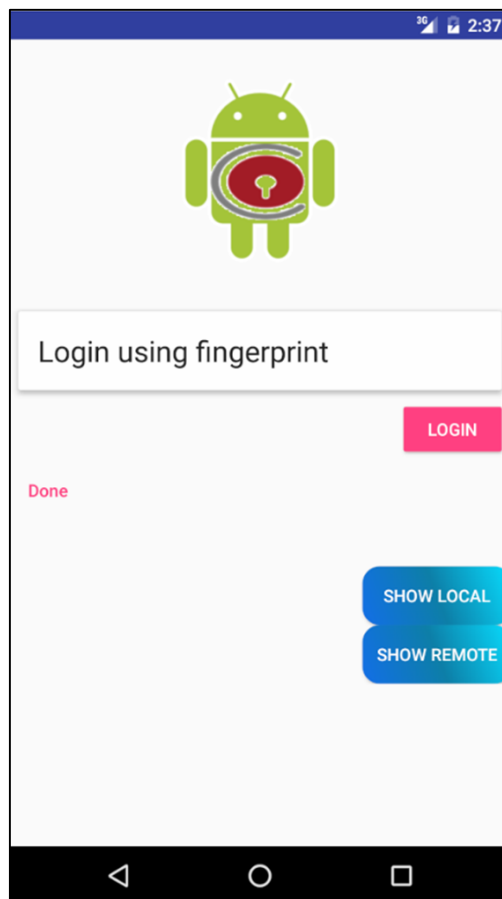


Figure 42: Credential Backup & Restore mobile application: Main page

3.1.7.2.2 Cryptographic Credentials stored on the device

The cryptographic credentials that are locally stored in the device are presented to the user in a list. Currently, they are fetched from a local SQLite database of the application. This SQLite database is also encrypted. Apart from the credentials data some metadata are also stored. The SQLite database

file is located in the folder of the device which is specific for this application’s identifier. Therefore, the way which credentials will be stored at the device has to be defined so that we can proceed with accessing them properly as ReCRED defines so. This has also to be unique and common between all ReCRED mobile sub-applications in order to have a common basis. For example, in order to use a cryptographic credential to prove an identity attribute to a Service Provider.

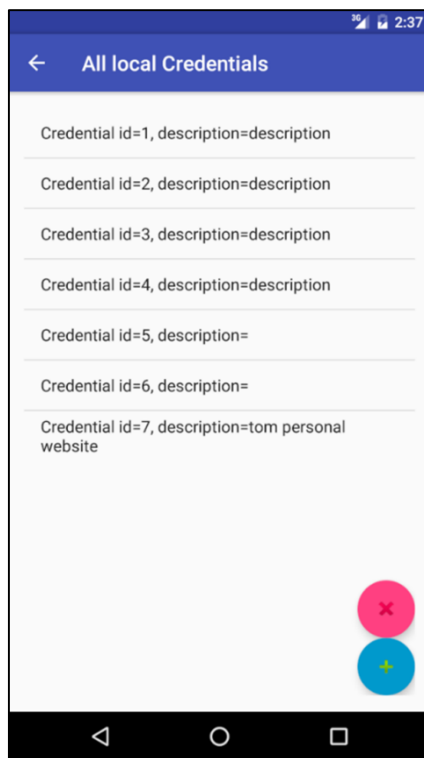


Figure 43: List with the Cryptographic Credentials stored in the mobile device

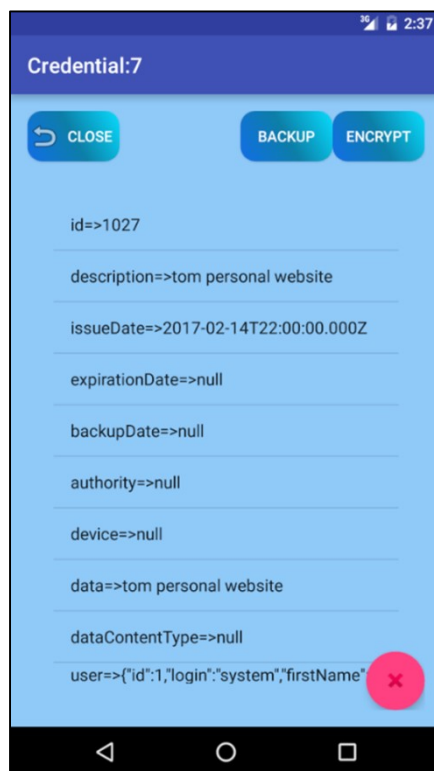


Figure 44: Page that shows to the user the details of a cryptographic credential

In the details of a locally stored cryptographic credential a user can see all the data and metadata of this credential. Additionally, the user can choose to encrypt the data and backup the credential to the Identity Consolidator server.

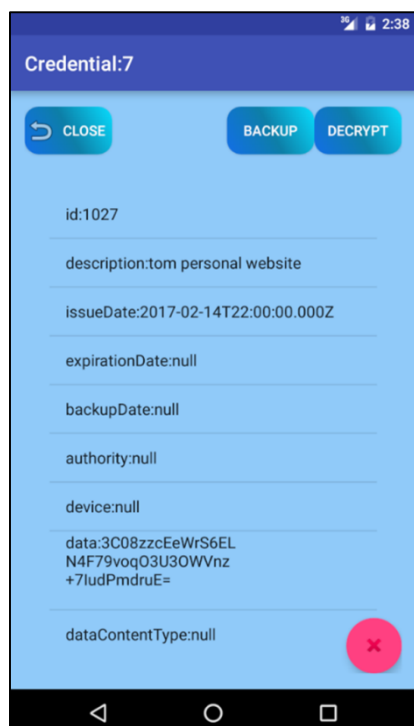


Figure 45: Details of an encrypted cryptographic credential

3.1.7.2.3 Cryptographic Credentials backup in the Identity Consolidator server

Currently, the details of the cryptographic credentials that are backed up in the Identity Consolidator server are presented to the user in a list. The user can choose which credentials to load and/or download and store to his mobile device.

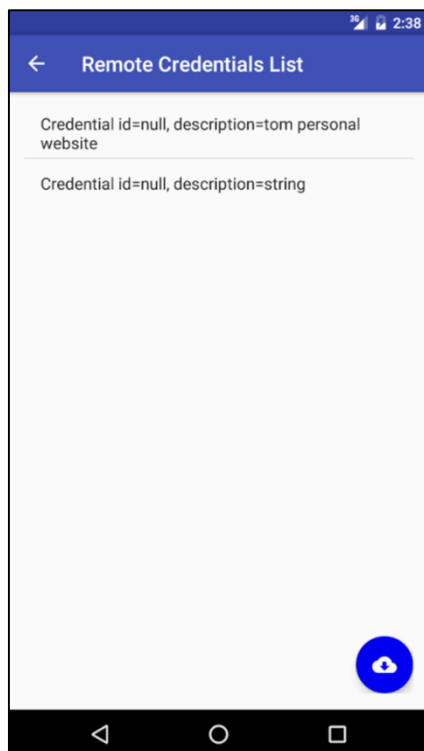


Figure 46: List with the Cryptographic credentials that are backed up in the Identity Consolidator

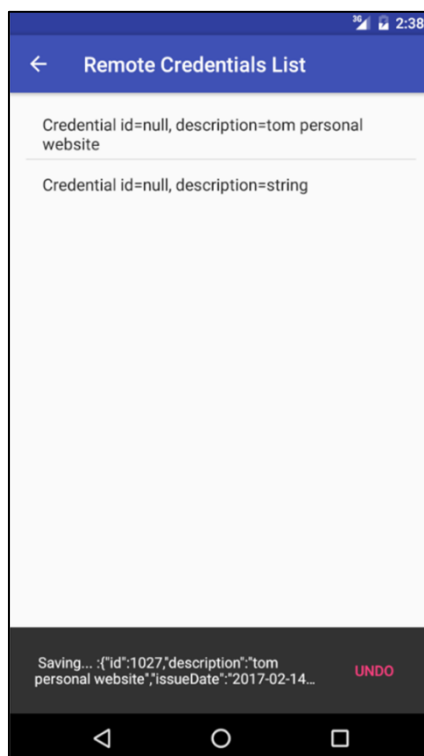
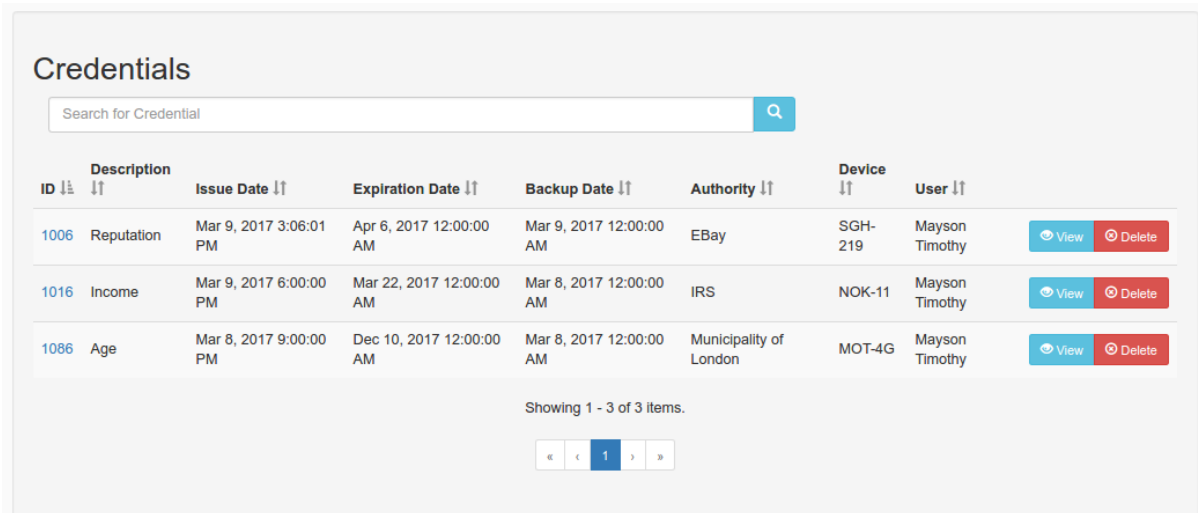


Figure 47: Process of fetching a cryptographic credential from the Identity Consolidator server

3.1.7.3 Web application

The web application offers a way for users to view their credentials in the Identity Repository over the web. The functionalities offered are only view, search, and remove. If a user has administrator privileges then the user may access all records, otherwise view, search and removed are constrained only to the user’s own credentials.



Credentials

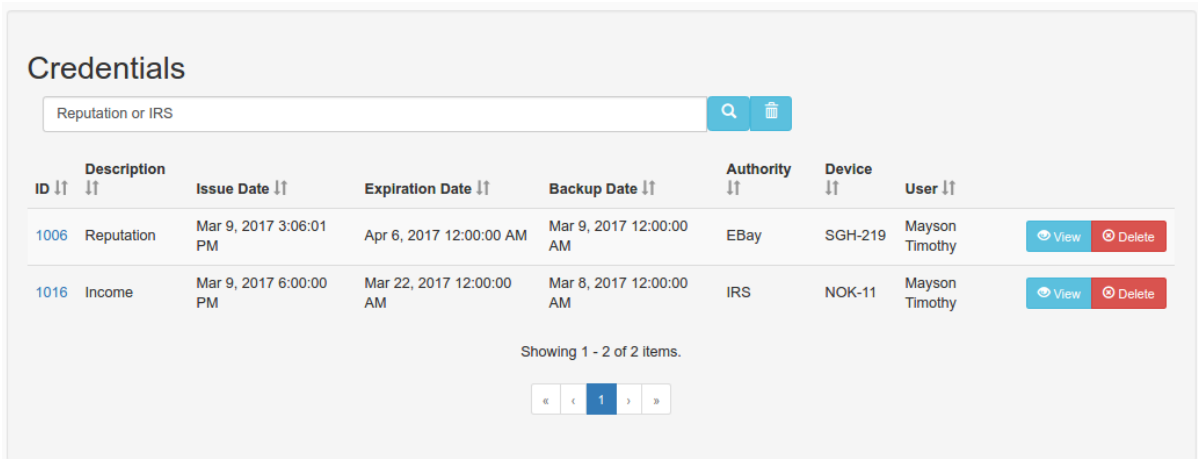
Search for Credential

ID ↑↓	Description ↑↓	Issue Date ↑↓	Expiration Date ↑↓	Backup Date ↑↓	Authority ↑↓	Device ↑↓	User ↑↓	
1006	Reputation	Mar 9, 2017 3:06:01 PM	Apr 6, 2017 12:00:00 AM	Mar 9, 2017 12:00:00 AM	EBay	SGH-219	Mayson Timothy	<input type="button" value="View"/> <input type="button" value="Delete"/>
1016	Income	Mar 9, 2017 6:00:00 PM	Mar 22, 2017 12:00:00 AM	Mar 8, 2017 12:00:00 AM	IRS	NOK-11	Mayson Timothy	<input type="button" value="View"/> <input type="button" value="Delete"/>
1086	Age	Mar 8, 2017 9:00:00 PM	Dec 10, 2017 12:00:00 AM	Mar 8, 2017 12:00:00 AM	Municipality of London	MOT-4G	Mayson Timothy	<input type="button" value="View"/> <input type="button" value="Delete"/>

Showing 1 - 3 of 3 items.

« < 1 > »

Figure 48: View list of user's backed-up credentials



Credentials

Reputation or IRS

ID ↑↓	Description ↑↓	Issue Date ↑↓	Expiration Date ↑↓	Backup Date ↑↓	Authority ↑↓	Device ↑↓	User ↑↓	
1006	Reputation	Mar 9, 2017 3:06:01 PM	Apr 6, 2017 12:00:00 AM	Mar 9, 2017 12:00:00 AM	EBay	SGH-219	Mayson Timothy	<input type="button" value="View"/> <input type="button" value="Delete"/>
1016	Income	Mar 9, 2017 6:00:00 PM	Mar 22, 2017 12:00:00 AM	Mar 8, 2017 12:00:00 AM	IRS	NOK-11	Mayson Timothy	<input type="button" value="View"/> <input type="button" value="Delete"/>

Showing 1 - 2 of 2 items.

« < 1 > »

Figure 49: Search user's credentials

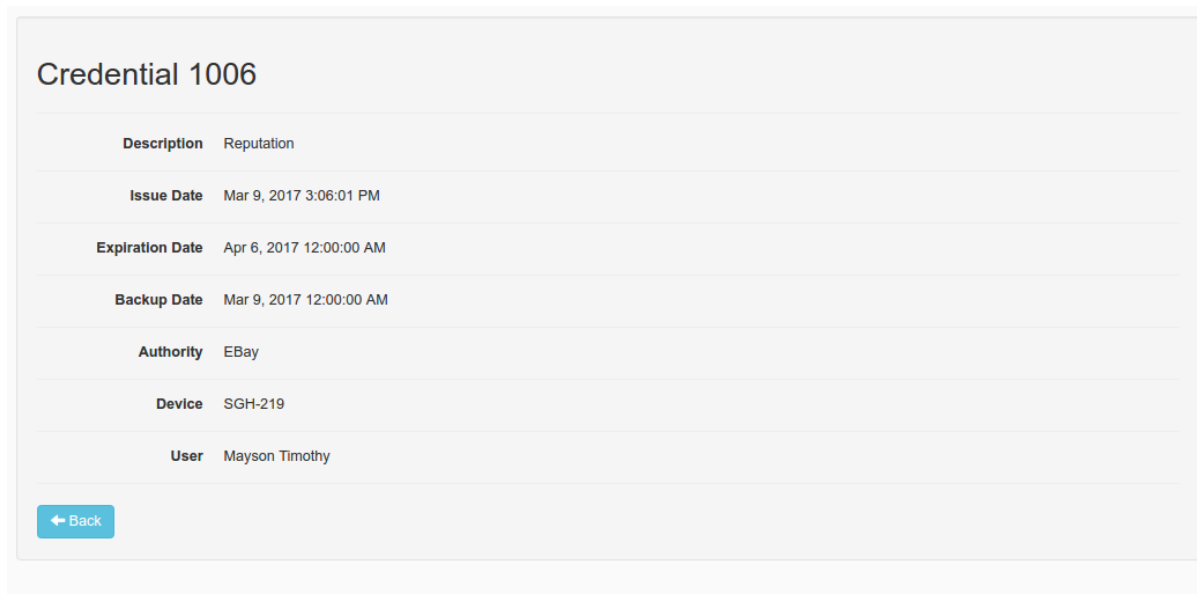


Figure 50: View the details of a credential

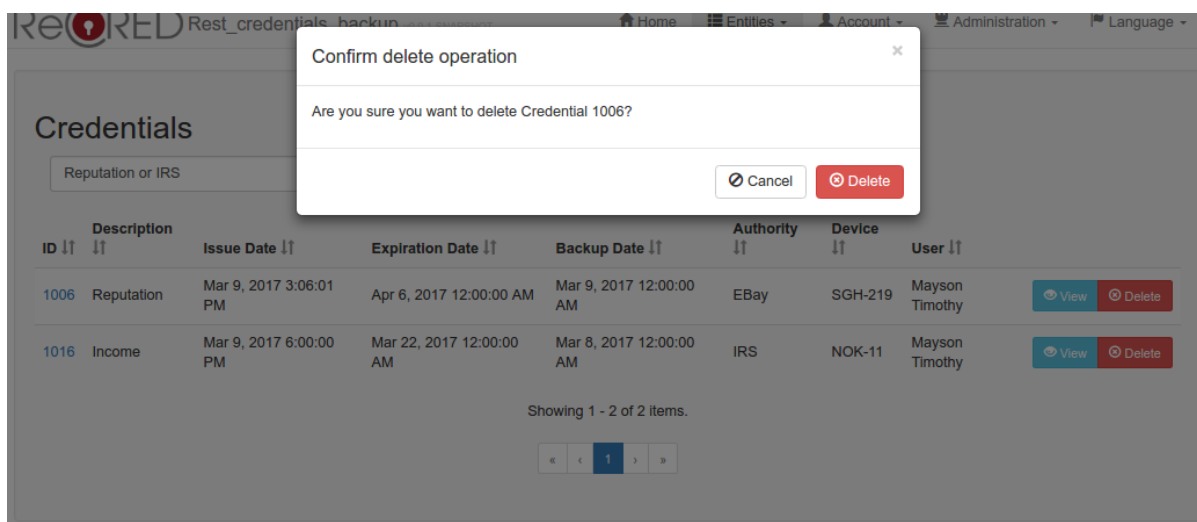


Figure 51: Delete a credential

3.1.8 OpenAM-based P-ABAC

In order to have a seamless integration between P-ABAC architecture and OpenAM infrastructure I is required to design a protocol able to mix the functionalities between the components of such two frameworks. Indeed, verifying IdPs should be able to seamlessly verify P-ABAC credentials and collaborate with OpenAM components to allow or deny users from accessing resources on Service Providers. To this aim we developed an OpenAM module able to understand the FIWARE Open RESTFUL API protocol. A very important requirement to be matched is the preservation of the anonymity of the user: it is an obvious feature in P-ABAC but it requires to be considered also in the integration with OpenAM protocols, where normally the identity of the user is known.

Note that an Identity Provider in OpenID Connect proves the identity of a user to a service, while an Identity Provider in P-ABAC is the issuer of the credentials. So we define:

- Verifying Identity Provider: the OpenAM module

- Issuing Identity Provider: the Idemix/U-Prove credentials issuer

The following figure shows the high-level flow of how a user can access a service provider using PABAC.

The diagram has following entities:

- Client: a client application to access the service of the service provider. This can, for example, be a mobile browser
- PABAC App: a mobile app that has cryptographic U-Prove and Idemix capabilities
- Service provider: a service (without FIWARE capabilities) that makes use of OpenAM
- Verifying IDP & FIWARE: The Verifying IDP based on OpenAM is closely integrated with FIWARE to verify end user credentials. This combination can, for example, be part of the ReCRED identity consolidator.

On a high level, the following steps are executed:

- The client tries to access the service
- The service provider launches an OpenID Connect authentication request to the verifying IDP
- The verifying IDP now needs to be able to launch the end user’s mobile app to proof the attributes. The mobile app needs to know the FIWARE endpoint, what to proof and also have a session identifier.
There are two ways to do this in a user-friendly way:
 - If the user makes use of a client on a mobile phone, the information is shown as a link that opens the application with the required parameters.
 - If the user client is not on a mobile device, a QR code is shown that contains the same link. The user can easily scan the code to open up the application.
- After showing the link or QR code, the client will start polling the verifying IDP. The verifying IDP validates with FIWARE, based on a session identifier, whether the attribute exchange was successful.
- The PABAC app can now execute the necessary protocols with FIWARE to proof the user’s attributes.
- Now, when the Verifying IDP checks with FIWARE again whether a particular session identifier has authenticated. As this is now the case, the Verifying IDP retrieves the attributes.
- Using standard OpenID connect mechanisms, the Verifying IDP can now send an authorization code to the service provider. The service provider can exchange this code with an access token and then finally retrieve the attribute.

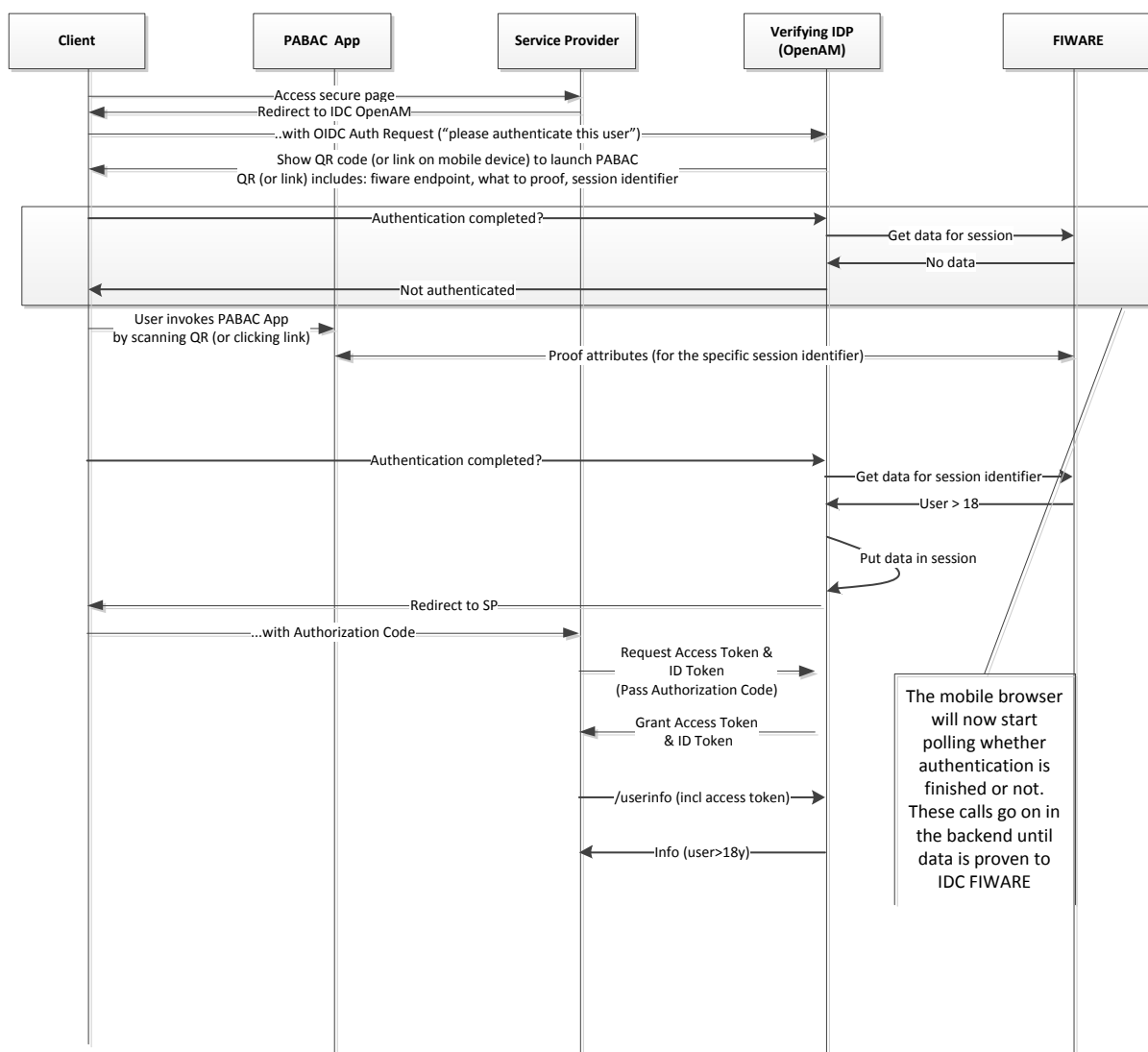


Figure 52 Protocol to integrate OpenAM within P-ABAC architecture

3.1.9 IRMA-FIWARE Integration

The integration of the IRMA implementation targeted to mobile devices (IRMA for short), described in Section 2.2.2.2, and the FIWARE Privacy Open RESTful API Specification (FIWARE for short), described in Section 2.2.2.1, aims at retaining the well-defined and general, support of multiple ABAC protocols of FIWARE while retaining the user-centric and device-centric approach of IRMA.

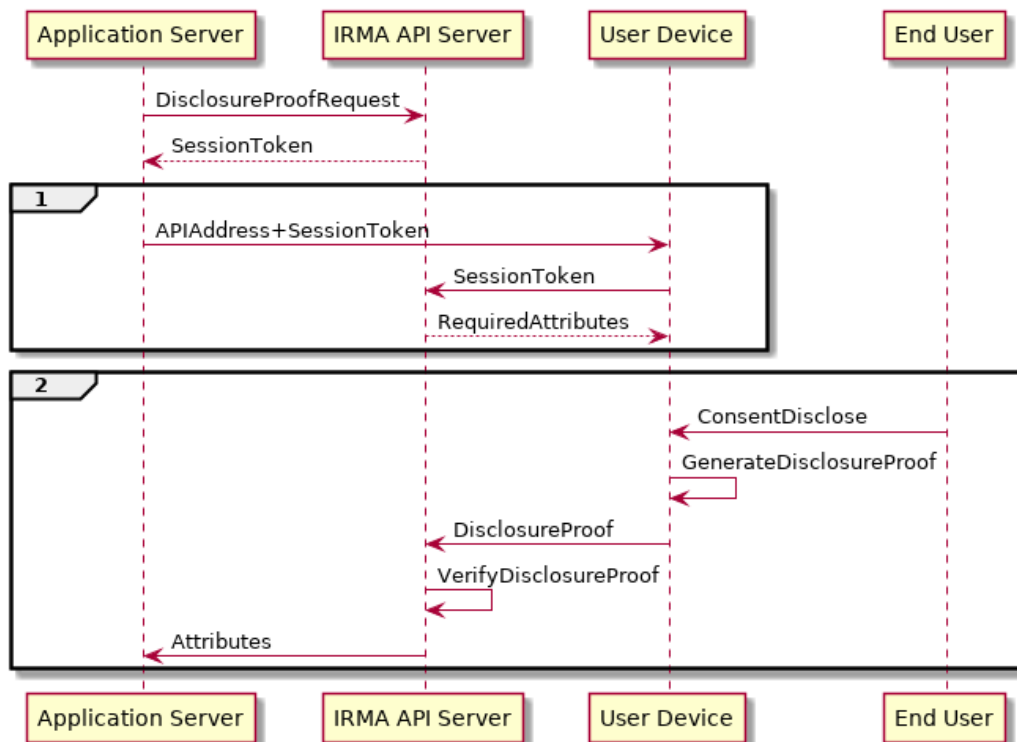


Figure 53 Mapping of the IRMA Verification Protocol into FIWARE Open RESTful APIs

Figure 53 summarizes the mapping of the IRMA verification protocol to FIWARE. A similar mapping is done for the issuing protocol.

Rectangle 1 contains the IRMA interactions that can be mapped to the FIWARE method `PresentationToken createPresentationToken(...)` in the interface associated to the User.

`RequiredAttributes` represents the data structure which is returned from the GET call towards the resource `/api/v2/verification/{verificationID}` of the IRMA Server and conceptually can be mapped on the structure `PresentationPolicy` specified by FIWARE.

`DisclosureProof` represents the data structure to be passed as parameter to the POST towards the resource `/api/v2/verification/{verificationID}/proofs` of the IRMA Server, and can be mapped on the FIWARE structure `PresentationToken`.

Rectangle 2 contains the IRMA interactions that can be mapped to the FIWARE method `PresentationTokenDescription verifyTokenAgainstPolicy(...)`.

`SessionToken` is the identifier of the ongoing verification session and one of them is created each time a Service Provider sends to the IRMA Server a `DisclosureProofRequest`. We thus map the `SessionToken` to its related `DisclosureProofRequest` and use it in the `PolicyUID` attribute when mapping the `DisclosureProofRequest` to the FIWARE `PresentationPolicy`. Moreover, to allow the adaptation, as `PolicyUID` is defined as being of type `AnyURI`, we can define it as `APIAddress+SessionToken`, as in the example below:

```
policyUID =  
https://<irmaservername>/api/v2/verification/<SessionToken>;
```

The `DisclosureProof` (which in the IRMA API server source code is called `ProofList` and is an array of “Proof” objects, which are defined in the IRMA “`credentials_idemix`” component), represents the list of cryptographic proofs related to `DisclosureProofRequest`. The `DisclosureProof`, as already mentioned is mapped to the structure `PresentationToken` defined by FIWARE.

As final remarks, we have to take into account that a critical point on the integration of the two protocols, IRMA and FIWARE, is that these are based on different architectures: IRMA relies on an intermediate entity, the IRMA API Sever, while FIWARE assumes the communications to occur end-to-end.

3.1.10 Attributes and Policies for P-ABAC

Policies in the P-ABAC infrastructure are managed by the ABAC reasoning tool. The tool’s principle function is the evaluation of resource requests based on existing policies. This is the policy decision point (PDP) which is structured based on the XACML format.

The reasoning tool core function revolves around the following process: A request is issued by a user to access a specific resource. The request includes the resource in question and the attributes that the user chose to reveal. Then the tool transforms the request in an XACML accepted format, evaluates the request based on existing policies (collectively stored in XACML format) and returns the decision.

Based on that core we have extended the reasoning tool in order to easily manage policies. These includes a web interface for the network administrator to create, view, delete policies and a policy recommendation system.

The rest part of this section is divided as follows. Subsection 3.1.10.1 will provide the network administrator’s point of view in creating policies. In subsection 3.1.10.2 we will take a closer look on the PDP. Subsection 3.1.10.3 will view the data collection of the Reasoning Tool and finally on subsection 3.1.10.4 the policy recommendation module.

3.1.10.1 Access Control Policy Management

The Access Control Policy Management module is a Web front end to facilitate the network administrator in creating Policies. Specifically, he/she will be able to manage Attribute-Based Access Control (ABAC) Policies and Account-Based Control (AccBAC) Policies.

Creating ABAC Policies is achieved by specifying the resource type and the relative attributes that should be present to permit its’ usage. Following is a screen shot of the Create ABAC Policy Tab.

ReRED Access Control Policy Management

Create ABAC Policy

Create AccBAC Policy

Show ABAC Policies

Show AccBAC Policies

Create Policy

Resource type:
Internet

Main Attributes

Title
Title

Department
Department

Personal Information

First name Last name Father's name Age Gender

First name Last name Father's name Age Gender

Date of birth Nationality Phone number Email address

Date of birth Nationality Phone number Email address

Figure 54 screen shot of the Create ABAC Policy Tab

After we have created our ABAC policies we can view them by accessing the Show ABAC Policies tab.

ReRED Access Control Policy Management

Create ABAC Policy

Create AccBAC Policy

Show ABAC Policies

Show AccBAC Policies

Policies

Policy	Resource
7	Internet
Title	
Professor	
Delete	

Policy	Resource
8	Webmail
Title	
Professor	
Delete	

Figure 55 Show ABAC Policies tab

Following is the create and view of AccBAC policies. AccBAC are policies aimed to give a specific account access to a resource.

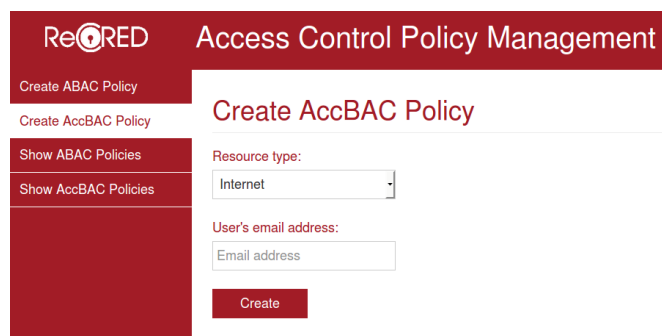
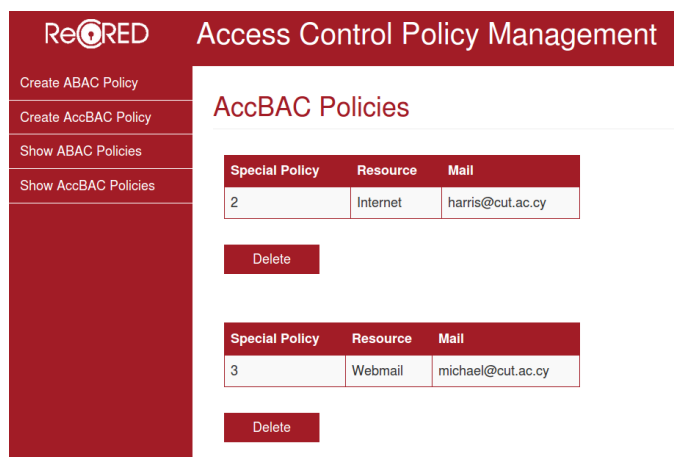


Figure 56 AccBAC Policies creation



Special Policy	Resource	Mail
2	Internet	harris@cut.ac.cy
Delete		
Special Policy	Resource	Mail
3	Webmail	michael@cut.ac.cy
Delete		

Figure 57 AccBAC Policies View

3.1.10.2 Policy Decision Point (PDP)

The PDP is the Reasoning Tool core function which evaluates the incoming requests based on the network administrator’s policies. The whole process is based on XACML an attribute-based access control policy language. Following is a typical example of the PDP evaluation.

First the request is received in a JSON format and is transformed to XACML acceptable JSON format. An example of the request is the following.

```
{
  "Request": {
    "Action": {
      "Attribute": {
        "AttributeId": "action-id",
        "Value": "access"
      }
    },
    "Resource": {
      "Attribute": {
        "AttributeId": "resource-id",
        "Value": "webmail"
      }
    },
    "AccessSubject": {
      "Attribute": [
        {
          "AttributeId": "department",
          "Value": "electrical engineering"
        },
        {
          "AttributeId": "title",
          "Value": "student"
        }
      ]
    }
  }
}
```

Figure 58 PDP Request Example

Then the request is evaluated against the policies that are collectively stored in an XACML format and the PDP returns the request decision (Permit/Denied). An example of a Policy is the following.

```
<Policy>
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">internet</AttributeValue>
          <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" Category="urn:oasis:names:tc:
        </Match>
      </AllOf>
    </AnyOf>
  </Target>
  <Rule>
    <Target>
      <AnyOf>
        <AllOf>
          <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">professor</AttributeValue>
            <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:accesssubject:title" Category="urn:oasis:names:tc:
          </Match>
        </AllOf>
      </AnyOf>
    </Target>
    <AllOf>
      <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">student</AttributeValue>
        <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:accesssubject:title" Category="urn:oasis:names:tc:
      </Match>
    </AllOf>
    <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Electrical Engineering</AttributeValue>
      <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:accesssubject:department" Category="urn:oasis:nam
    </Match>
  </AllOf>
</Rule>
</Policy>
```

Figure 59 PDP Policy Example

3.1.10.3 Data Collection

The Access Control Reasoning Tool uses a database to store information on policies and requests. The policies are stored in order to properly manage them through the access control management module. And the requests are stored in order to keep a log of the requests but also to be used in the policy recommendation system.

Following are examples of the database tables, policies and requests.

#	id_abac_policies	resource	firstname	lastname	age	nationality	religion	country	city	phone	email	title	department	yearofstudy	semester	teachingyears	scholarship
1	7	internet	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	professor	NULL	NULL	NULL	NULL	NULL
2	8	webmail	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	professor	NULL	NULL	NULL	NULL	NULL
3	9	moodle	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	professor	NULL	NULL	NULL	NULL	NULL
4	10	research network	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	professor	NULL	NULL	NULL	NULL	NULL
5	11	cut apps	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	professor	NULL	NULL	NULL	NULL	NULL
6	12	file server	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	professor	NULL	NULL	NULL	NULL	NULL
7	13	print	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	professor	NULL	NULL	NULL	NULL	NULL
8	14	internet	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	student	Electrical Engineer...	NULL	NULL	NULL	NULL
9	15	webmail	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	student	Electrical Engineer...	NULL	NULL	NULL	NULL
10	16	moodle	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	student	Electrical Engineer...	NULL	NULL	NULL	NULL
11	17	research network	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	student	Electrical Engineer...	NULL	NULL	NULL	NULL
12	18	cut apps	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	student	Electrical Engineer...	NULL	NULL	NULL	NULL
13	19	File Server	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	student	Electrical Engineer...	NULL	NULL	NULL	NULL
14	20	print	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	student	Electrical Engineer...	NULL	NULL	NULL	NULL

Figure 61 Database table: policies

#	log_id	attribute_name	attribute_value
17	6	response_abac	permit
18	6	title	student
19	7	department	electrical engineering
20	7	response_abac	permit
21	7	title	student
22	8	department	electrical engineering
23	8	response	denied
24	8	title	student
25	9	department	electrical engineering
26	9	response_abac	permit
27	9	title	student
28	10	department	electrical engineering
29	10	response	denied
30	10	title	student
31	11	department	electrical engineering

Figure 60 Database table: requests

3.1.10.4 Policy Recommendation System

The policy recommendation system is a collection of algorithms which is responsible for making recommendations to the network administrator to improve the current policies. Major goals include improving ABAC policies (e.g. bundle policies, merge policies, delete obsolete policies) and decrease of AccBAC Policies by replacing them with ABAC policies.

Recommendation policies algorithms are deployed under two settings. The first setting is an automated process executed when a new policy is created and the second is initialized by the network administrator.

When a new policy is created is firstly checked for duplicates and then for redundancies. If a duplicated is discovered, then the policy should not be created. If a redundancy is discovered, then a recommendation to the network administrator is presented to choose an appropriate action.

The redundancy can be of two types. The first type of redundancy occurs when the policy is already covered by a more general policy. In this case the recommendation for the network administrator will be to remove the new policy. The second type occurs when the policy is a more general policy of a subset of already defined policies. This makes the other policies obsolete and a recommendation to the network administrator is presented to remove the obsolete policies.

The network administrator also has the option to request policy recommendations from the system. These policy recommendations can be derived from various methods. One of them is completeness in existing policies. For example:

“if a user is student and male”, he can get access to the Internet

“if a user is student and female”, she can get access to the Internet

The previous two statements are complete when all options of the sex attribute are presented and can be replaced with

“if a user is student”, he/she can get access to the Internet

Here the system will recommend to the network administrator that the two first policies can be replaced with the later policy. This bundles an existing subset of policies into one.

The second method recommends policies based on partially complete policies. This effectively recommends to the network administrator a policy with a high probability of being a valid one.

The third method makes recommendations based on requests logs. The logs contain information that can be used to make policy recommendations. Specifically, we are using generative models (Restricted Boltzmann Machines and Variational AutoEncoder) to model the hidden distribution of the data. This allows us to generate new policies by sampling the hidden distribution.

3.1.11 ABE-Based P-ABAC Solution for Wi-Fi

The work reported in this section has been presented at “Workshop on Hot Topics in Planet-scale mObile computing and online Social neTworking” (HotPOST) 2016 [69]. It shows how Attribute-Based Encryption (ABE) can be used for P-ABAC in a practical scenario.

Two mainstream techniques are traditionally used to authorize access to a WiFi network. Small scale networks usually rely on the offline distribution of a WPA/WPA2 static pre-shared secret key (PSK); security hence relies on the fact that this PSK is not leaked by end user, and is not disclosed via dictionary or brute-force attacks. On the other side, Enterprise and large scale networks typically employ online authorization using an 802.1X-based authentication service leveraging a backend online infrastructure (e.g. Radius servers/proxies). In this work, we propose a new mechanism which does not require neither online operation nor backend access control infrastructure, but which does not force us to rely on a static pre-shared secret key. The idea is very simple, yet effective: directly broadcast in the WLAN beacons an encrypted version of the secret key required to access the WLAN network, so that only the users which possess suitable authorization credentials can decrypt and use it. This proposed approach clearly decouples the management of authorization credentials, issued offline to the authorized end users, from the actual secret key used in the WLAN network, which can thus be in principle changed at each new user’s access. The solution described in the paper relies on attribute-based encryption, and is designed to be compatible with WPA2 and deployable within standard 802.11 management frames. Since no user identification is required (access control is based on attributes rather than on the user identity), the proposed approach further improves privacy. We demonstrate the feasibility of the proposed solution via a concrete implementation in Linux-based devices and via relevant testing in a real-world experimental setup.

With the increasing number of smart mobile devices, mobile users are willing to have ubiquitous access to the Internet. As predicted by Cisco [17], the number of personal mobile devices will grow to 8.2 billion by 2020. However, the existing cellular network (i.e., 3G, 4G, LTE) is not able to support this growing demand of mobile users. As a consequence, the widely available WiFi systems are considered to be the main choice for offloading the data traffic [18][37]. Cisco [17] predicted the amount of offloaded traffic from 3G and 4G to increase to 48% and 58%, respectively, by 2020. However, open un-protected WLANs, deployed in several public locations, are vulnerable against security attacks [34], for which several protocols have been designed in order to tackle this issue [24].

3.1.11.1 Introduction

An important challenge in using WiFi connections is to provide a secure and convenient way for user authentication and access control. Typically, upon connecting to an open WiFi network, the user is presented with a splash page that requires to authenticate or register. On the other side, access control in protected WLANs is non-trivial: the user needs credentials, which have to be obtained through another channel or offline, to connect to a protected network. Although traditional security protocols, i.e., Wired Equivalent Privacy (WEP) and WiFi Protected Access (WPA) are prone to several security attacks [24][27], these can be prevented by employing WPA2, with e.g. IEEE 802.1X [16]. However, when a service provider deploys such advanced authentication mechanisms, or even Radius-based authentication federations, it is required to setup and maintain online, interactive authentication infrastructures. In this scenario, an untrusted WiFi Access Point (AP) might threaten users' privacy, since a curious service provider would be able to track the clients connected to the APs [26].

We believe that attribute-based access control (ABAC) [32] is a promising solution for providing secure, privacy-preserving authentication and access control in such scenarios. ABAC is an access control mechanism in which the access of the users to a specific content/resource/object is specified based on the attributes of the user (e.g., occupation). The main advantage of ABAC compared to the other access control mechanisms, such as role-based or identity-based access control, is its flexibility especially in dynamic access control decisions, where there is no a priori information about the users, and large scale scenarios[31].

Recently, several researchers tend to adopt Attribute-Based Encryption (ABE) [35] to provide privacy and ABAC solutions in several scenarios, such as online social networks [22], cloud computing[33], and location-based services [28]. ABE provides fine-grained access control over data through defining attribute-based access policies. In particular, Ciphertext Policy Attribute-Based Encryption (CP-ABE) [23], which is an instantiation of ABE, allows the data owner to encrypt data specifying expressive access control based on a set of attributes. Only the users who have the right attributes in their decryption keys, will be able to decrypt the ciphertext. In this section, adopting CP-ABE, we propose WI-FAB, an Attribute-Based WLAN access control mechanism, without pre-shared keys and backend infrastructures. In our proposed approach, we introduce a clear separation between the authentication and issuance infrastructure, and the Authorization infrastructure. In particular, we encrypt the WPA2 secret, utilized to secure the WiFi connection, using CPABE and then divide it into several chunks. We then insert each chunk in the WLAN beacons and broadcast it in the network. Only the users who can rebuild the information included in beacons and decrypt it, and hence can retrieve the WPA2 secret, are authorized to connect to the network. To the best of our knowledge, the

proposed approach is the first that does not require any pre-shared key. Through extensive experimental results, we show that WI-FAB is secure, efficient and scalable.

3.1.11.2 Background

In this section we provide background knowledge on the concepts that we adopt in our proposed approach.

3.1.11.2.1 Attribute-Based Encryption

Attribute-Based Encryption (ABE) [35], is a powerful public key encryption scheme, in which encryption and decryption are based on descriptive attributes (such as age, gender, or occupation). Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [23] and Key-Policy Attribute-Based Encryption (KP-ABE) [30] are the two main types of ABE. In CP-ABE, the data owner enforces an access policy on the ciphertext. A user will be able to decrypt the ciphertext, if and only if, her decryption private key satisfies the defined access policy. While, in KP-ABE the access policy is bound to the decryption key of the user. She is able to decrypt a ciphertext if the attributes specified on the ciphertext matches her key’s access policy. Since CP-ABE provides the data owner with a means to have more control over the data, more researchers have concentrated on adopting CP-ABE in several applications [36]. In the following we provide explanation of the CP-ABE basic functions:

- Setup. Taking a security parameter as input, it outputs the public parameter PK, and a master key MSK.
- KeyGen. Taking a set of attributes SU, the master key MSK and the public parameter PK, it outputs a decryption key SKSU reflecting the given attributes.
- Encryption. Taking as input a message M, an access policy Π , and the public parameter PK, it outputs the ciphertext E.
- Decryption. Taking as input the ciphertext E that is encrypted under the access policy Π , the decryption key SKSU, and the public parameter PK, it outputs the message M if and only if SU “satisfies” the access policy Π .

ABE has several advantages compared to the other public key encryption methods [20]: (i) ABE provides fine-grained access control over data through allowing the data owner to define expressive access policies based on the attributes; (ii) the proposed approaches based on ABE are scalable and independent of the number of authorized users; (iii) ABE is efficient in terms of communication, storage and key management overhead. This is due to the fact that there is no need for sharing any secret between the parties.

3.1.11.2.2 Attribute-Based Access Control

Attribute-based access control (ABAC) [32][29] is a flexible access control method in which the acceptance or rejection decision for accessing a resource is made based on the attributes of the requester. ABAC is indeed efficient in terms of communication overhead between the requester and the resource owner. This is due to the fact that the two parties do not need to agree on a pre-shared key to access the resource. Moreover, ABAC preserves the privacy of the users in the sense that the access credentials are not bound to the user identity. Instead, the resource owner only defines the access policies for the resource, and the user will be authorized to access the resource if and only if her credentials, which are bound to her attributes (such as citizenship or group membership) satisfy the access policy.

3.1.11.2.3 WPA2 Protocol

The Wi-Fi Protected Access 2 (WPA2) protocol [19] is a rectification of the 802.11 standard, which is introduced in order to address the security vulnerabilities of the Wi-Fi Protected Access (WPA) protocol for wireless networks. WPA2 supports the use of Advanced Encryption Standard (AES) in order to provide data confidentiality and integrity. Moreover, WPA2 provides both personal and Enterprise authentication capabilities [21]: in the personal authentication method, WPA2 makes use of Pre-Shared Key (PSK), while, in the Enterprise mode, the users need to be authenticated based on the IEEE 802.1X.

3.1.11.2.4 IEEE 802.11 Beacon Management frames

The IEEE 802.11 standard [15] defines several subtypes of management frames. Among these, Beacon frames are broadcast periodically by the access point to advertise its presence, provide the SSID (i.e. the name of the wireless network) and announce its capabilities and other parameters to other wireless devices within its range. These data included in Beacons are enclosed in a sequence of field tuples called Information Elements. Of specific interest for this work are Vendor-Specific Information Elements, which are used to carry information which is not explicitly defined in the IEEE 802.11 standards.

3.1.11.2.5 Fountain Coding

Digital fountain (also known as fountain coding) first introduced in 1998 in order to provide a reliable distribution of bulk data to a large number of users [25]. Fountain coding allows a data owner, who wishes to send a data consisting of a sequence of m equal length packets, to send a stream of distinct packets (called encoding packets or droplets) into the network. The receiver will be able to reconstruct the source data by receiving any subset of the encoding packets composed of exactly m number of packets. Fountain coding is reliable, in the sense that it guarantees all the intended users will receive the data source. Moreover, it provides an efficient and on-demand method of sending data to the user in a lossy environment.

3.1.11.3 Proposed Approach

We have devised a privacy-preserving attribute-based access control mechanism for resources which are protected by a shared secret, leveraging the CP-ABE scheme proposed in [23]. Figure 62 shows an overview of our proposed approach. We use CP-ABE to encrypt a secret that grants a single access to a resource.

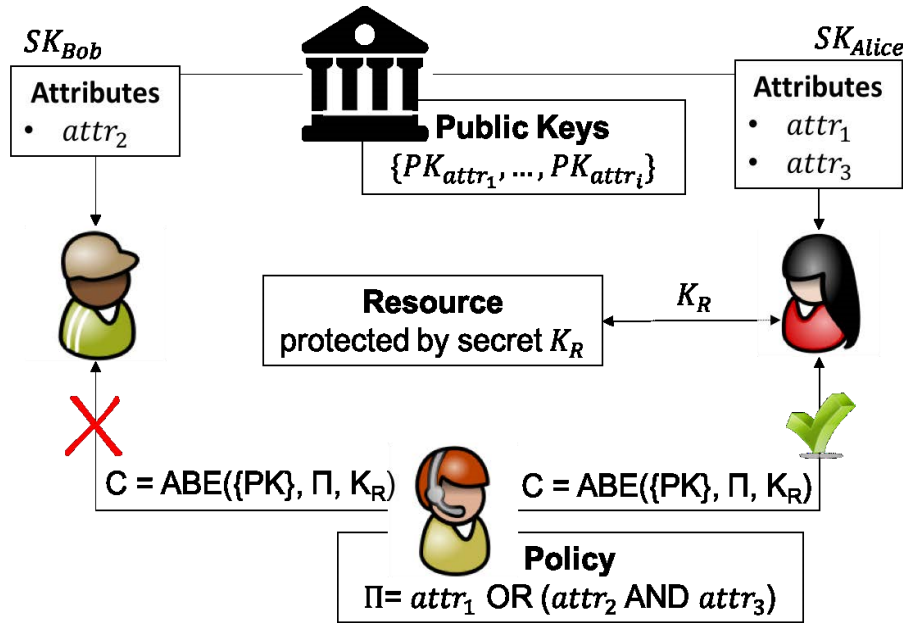


Figure 62 CP-ABE based proposed ABAC mechanism

An authority A generates a master key MSK, a public key for each attribute it wishes to support, and also a key SKSU, associated to a set of verified and appropriate attributes SU, for each user U.

Let K_R be the secret that protects a single access to a resource R, and Π the attribute-based policy that a user V holding R wants to enforce on the access to this resource. Then V encrypts K_R using CP-ABE according to the policy Π and publishes the encrypted secret $C = ABE(\Pi, K_R)$.

The users which own the keys whose set of associated attributes satisfy the policy Π can successfully decrypt C and obtain K_R . K_R can subsequently be employed to access the resource R once.

3.1.11.3.1 WI-FAB

We employ the above-described access control mechanism in a scenario in which the resource is a wireless network protected by a WPA2 secret. We call the below-described system WI-FAB, outlined in Figure 63. For each user U, an authority A, which can verify a set of attributes SU of U, releases a private key SKSU. A WLAN service provider W that trusts the authority A, for each of its access points AP in which it wants to enforce the policy Π :

1. generates a new random secret KW
2. sets KW as the WPA2 secret of the AP
3. encrypts KW using CP-ABE and the policy Π , yielding $C = ABE(\Pi, KW)$
4. sends C to nearby users in the IEEE 802.11 information elements of the beacons
5. when a user successfully connects to AP, the here described procedure starts again from point 1.

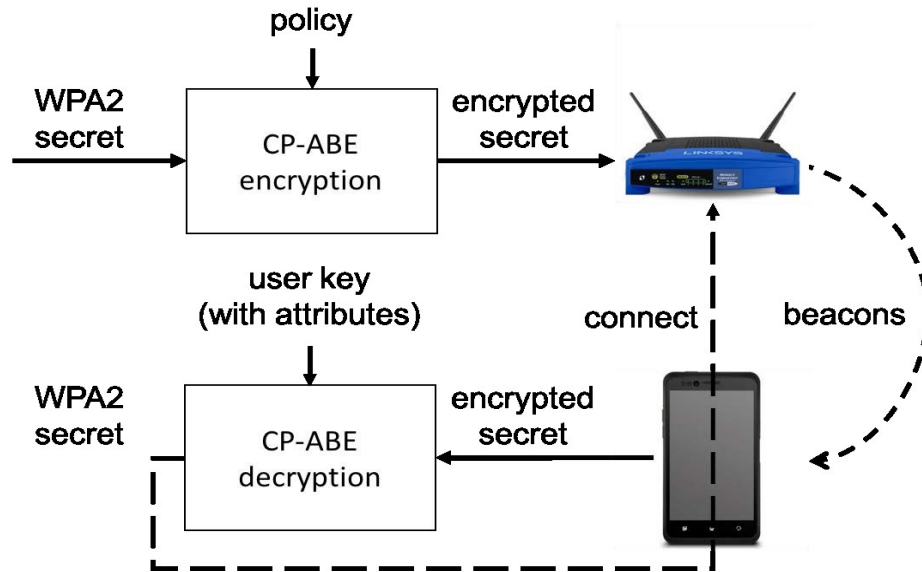


Figure 63 WI-FAB overview diagram

A user U who wishes to access the WLAN service provided by W :

1. captures beacons (by scanning or sniffing) from the access point AP
2. extracts the IEEE 802.11 information elements which contain C
3. attempts to decrypt C using her key SK_{SU}
4. if the decryption is successful, i.e. if the attributes in SU satisfy the policy Π , then U holds KW
5. U generates a random MAC address for her wireless interface
6. U employs KW to generate a WPA2 configuration and connects to AP using the WPA2 protocol

Note that KW is used by the WPA2 protocol to generate a per-client session key. Thus, when W generates a new secret for an access point and sets it as the new WPA2 secret, the users which are already connected to that access point are not disconnected. Moreover, the transmission of $C = ABE_{\Pi}(KW)$ in the IEEE 802.11 information elements of the beacons can be encoded using fountain coding (§ Section 2). In this way if the size of C exceeds the maximum size of a single information element, it can be encoded into smaller droplets. The user can obtain C by capturing enough of these droplets and using fountain (de)coding. Figure 64 sketches the proposed format of the Information Elements contained in the beacons emitted by the access point. The first fields are employed as specified in the IEEE 802.11 standard [15], i.e. the Element ID (one octet) contains the value 221, assigned to Vendor-Specific Element IDs; the field length (one octet) is set to the aggregated size of the subsequent fields; and OUI (three octets) should contain an identifier assigned by IEEE, but for experimentation purposes is temporarily set to an arbitrary (unassigned) value. The index field (one octet) is incremented modulo 256 each time that the AP changes the WPA2 secret, and is used by Stations to discard collected droplets associated to WPA2 secrets which become invalid. The n chunks (one octet) and seed (two octets) fields contain respectively the total number of chunks in which C has been divided, and the pseudo-random seed used for the current droplet. Finally, the data field (variable length) contains the actual droplet data.

0			31
Element ID	length	OUI	
OUI	index	nchunks	seed
seed	data (variable length)		

Figure 64 Format of the Vendor Specific Information Elements included in the IEEE 802.11 beacons broadcast by the access point

3.1.11.4 Implementation

We have performed a Linux-based preliminary implementation of our proposed system, summarized in Figure 65. Although this implementation has not yet been tested on actual embedded and mobile devices, we foresee no big obstacles to its porting to at least other Linux-based operating systems such as OpenWrt/LEDE for the AP part, and Android for the STA part.

Note that our implementation did not require changes to the Linux kernel, but only to userspace tools such as hostapd and iw, integrated with an implementation of a custom variant of the fountain LT codes and bash scripts. For the CPABE part, we are employing the implementation of [23] provided at [11]. Further details are given in the remainder of this section.

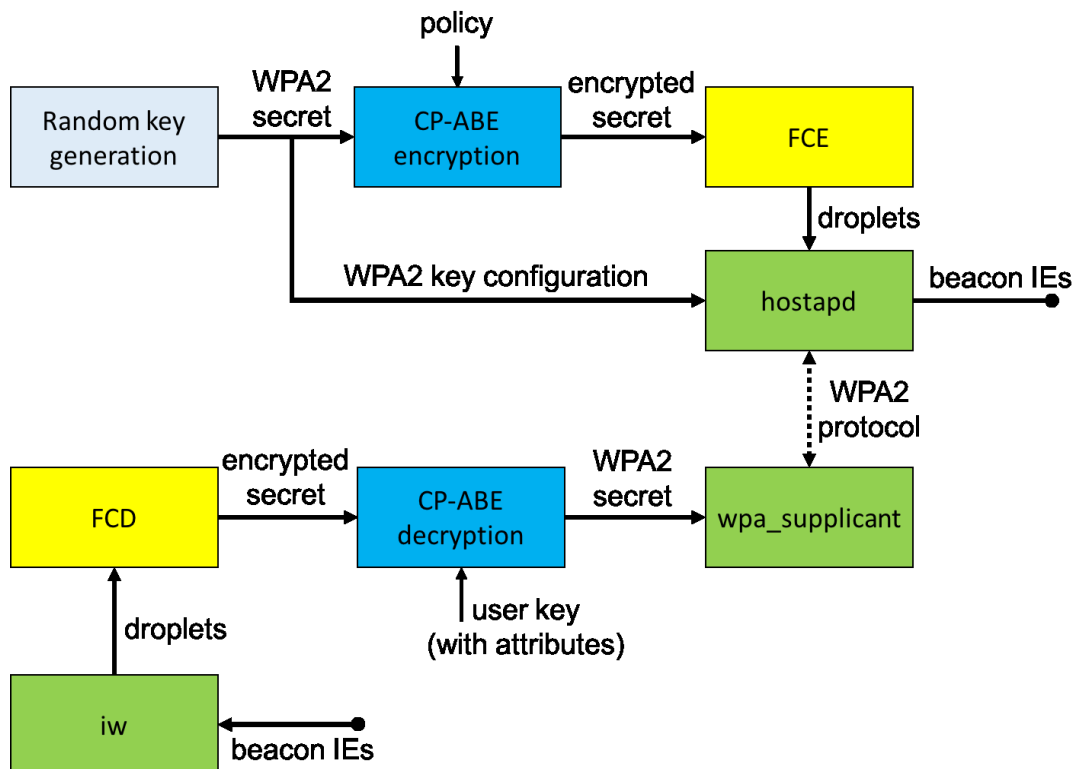


Figure 65 WI-FAB implementation overview

3.1.11.4.1 Fountain Coding

We call FCE and FCD the implementations of the fountain coding encoder and decoder. FCE continuously reads the contents of a specific file and provides droplets to a configured named pipe.

Droplets include an index number, as described in Section 3.1. When FCE is instructed to reload the contents of the file, the index is increased. FCD performs the inverse operation: it continuously reads droplets from a named pipe and when is able to reconstruct the original information it writes it to a file.

3.1.11.4.2 Access Point

The `hostapd` daemon is a highly configurable user space IEEE 802.11 access point implementation that employs the `nl80211` Linux API. Its source code is publicly available at [12]. Please note that `hostapd` already provides a mean to configure static Information Elements to be included in the access point beacons. Our modifications to `hostapd` allow to: 1) provide a dynamic stream of Information Elements through a local named pipe, and 2) change the configuration of the WPA2-PSK keys without restarting the demon and without disconnecting the users that are already connected. We have devised and implemented a bash script that periodically:

- generates a new random 256 bit WPA2 key `S`
- encrypts `S` with CP-ABE using a configurable policy and the supplied CP-ABE public key and stores it in a file `F`
- configures `S` as a new WPA2-PSK secret on `hostapd`
- instructs FCE to reload `F` and to provide its droplets to `hostapd` (through the named pipe).

In order to allow a reasonable time for the connection of the users, we keep the last two generated WPA2-PSKs active at the same time. This is possible as `hostapd` supports the use of multiple coexistent WPA2-PSKs.

Note that the above implementation is only an approximation of the scheme described in Section 3, as the WPA2 key is changed at regular intervals instead of being changed each time a user successfully connects to the AP.

3.1.11.4.3 Station

`iw` is a configuration utility for wireless devices. Its source code is publicly available at [13]. Among its features, it allows to instruct the wireless driver to perform active and passive scans and to output the results. We have performed minor modifications to `iw` to change the way in which the IE contained in the beacons are displayed. In the future we plan to employ `wpa_supplicant` [14] instead, which is a standard tool in Linux-based OS and Android OS, responsible for the connection to IEEE 802.11 WLANs.

We have developed a bash script that runs `iw` based active scans multiple times and sends its output through a named pipe to FCD. When FCD has collected enough droplets to reconstruct the contents of `F`, a decryption attempt, using CP-ABE and the configured secret key of the user, is made. If the decryption is successful (i.e. if the supplied secret key of the user is associated to a set of attributes that satisfies the policy configured at the access point), the decrypted WPA2-PSK secret is included in a `wpa_supplicant` configuration file and a connection to the access point is performed, without the need to interact with the user.

3.1.11.5 Results

This section shows the effectiveness of the proposed solution in a real-world experimental setup in our lab. We are employing laptops (on the same desk) with a Linux-based OS running the implementation described in Section 4. The wireless NICs employ the `rt2800usb` driver module. The version employed for the CP-ABE tools is 0.11, while our modified `hostapd` and `iw` are derived from

versions 2.5 and 3.3, respectively. Unless explicitly stated, we have configured at the AP a beacon interval of 102.4 ms (i.e. the default value of hostapd) and a policy with four attributes.

In the first experiment, we demonstrate the effectiveness of fountain coding as opposed to the approach of just dividing the encrypted secret into numbered chunks.

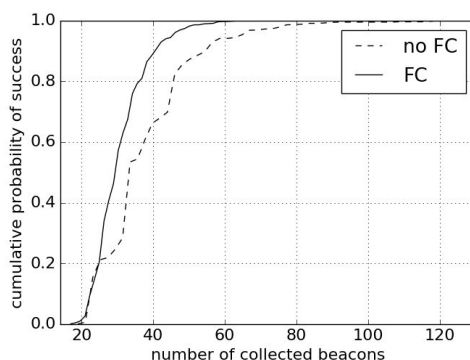


Figure 66 ECDF associated to the number of collected beacons needed to reconstruct the encrypted secret with and without fountain coding (FC)

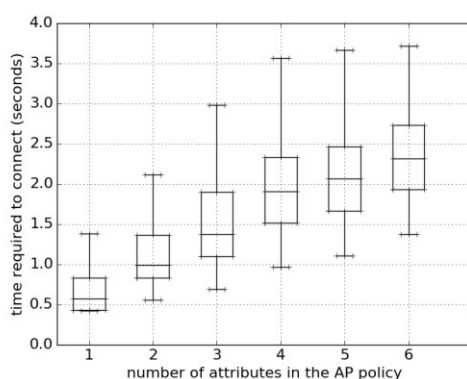


Figure 67 Time needed for the station to connect vs. number of attributes in the AP policy

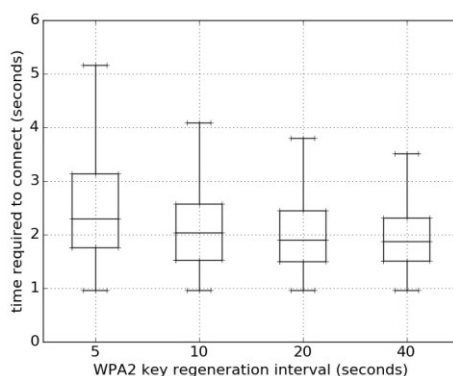


Figure 68 Time needed for the station to connect vs. random WPA2 key regeneration interval (500 connection attempts for each regeneration interval, policy with 4 attributes)

Figure 66 shows the empiric probability (on 1000 samples, i.e. station connection attempts) of recovering the encrypted secret (divided into 16 chunks) vs. the number of collected beacons. Beacon collection make up the majority of the total time needed by the station to successfully connect to the

AP. Although our fountain coding implementation has room for substantial improvement, the performance increase of the system is promising.

In the second experiment, we change the size of the policy by varying the number of contained attributes. This change the size of the encrypted secret and thus the time needed for the station to connect to the AP.

Figure 67 shows how the number of attributes affects the performance, for a series of 500 connection attempts for each set number of attributes. The size of the encrypted secret depends on the number of attributes employed in the policy. In the third experiment we modify the WPA2 key changing interval and observe how this change affects the time needed by the stations to connect to the AP. In Section 3.1 we propose to change the WPA2 secret each time a new station successfully connects to the AP. The results shown in Figure 68 support the feasibility of this proposal.

3.1.11.6 Conclusions

We introduced a new attribute-based access control mechanism based on CP-ABE, called WI-FAB. We proposed to employ our mechanism in the scenario of protected WLANs and we implemented a working prototype. We finally showed and discussed the results obtained in a real-world experimental setup.

In the future, we plan to extend the proposed approach to leverage the multi-authority CP-ABE schemes. This will enable federation scenarios in which multiple entities will be able to issue attribute-based credentials to users. Moreover, we plan to investigate further the use of fountain codes, optimizing their usage, the implementation and the employed parameters, as well as other alternative error correction mechanisms.

3.2 P-ABAC Components Mapping

The modules described in the above sections can be mapped to the main P-ABAC components, i.e. User (Prover), Issuer and Verifier. The FIWARE Privacy Open RESTful API provides a common interface for all the Idemix and U-Prove modules. The mapping is provided in the table below.

Table 4 P-ABAC Modules to Components Mapping

P-ABAC Component	Module	Section	Function
User (Prover)	ReCRED Wallet mobile app	Section 3.1.1.2	Management of the P-ABAC Credentials
User (Prover)	Backup mobile app	Section 3.1.7	P-ABAC Credentials Backup
User (Prover)	Consent Management mobile app	Section 3.1.4	Manage the consent management, i.e. define user-side policies on attributes
User (Prover)	Trusted Execution Environment (TEE)	Section 3.1.3	Protect the secret parameters of the P-ABAC protocols
User (Prover)	De-anonymization risk assessment mobile app	Section 3.1.5	Provide to the user an estimation of their probability of being deanonymized
User (Prover)	P-ABAC and FIDO Integration	Section 3.1.6	Provide a means to use P-ABAC credentials over the FIDO Protocol
Issuer	Credential Management Daemon	Section 3.1.1	Issue P-ABAC credentials

Verifier	OpenAM-P-ABAC	Section 3.1.8	Provide a means to verify P-ABAC attributes over the OpenID Connect protocol
Verifier	Attributes and Policies for P-ABAC	Section 3.1.10	Policy definition, enforcement and recommendation

4 Application Scenarios

4.1 Support to Financial Services

The financial services support pilot focuses on the loan-origination use case. Bank clients requesting a banking product e.g. credit card or consumer loan, are required to present information including personal, professional, financial and other details to the banking institution, depending on which their request will be approved or rejected.

With the current infrastructure, anonymity is by default forfeited. Clients have to manually collect all the necessary documents and, on top of that, they reveal unnecessary personal details because the submitted documents include them by design. Finally, extra verification is necessary in order to ensure the authenticity of the submitted documents, and additional time will be spent profiling and scoring the client. This time-consuming and paperwork-intensive process can be vastly improved using the ABAC architecture.

Every piece of information that the banking institution requests can be individually revealed and certified by the corresponding authority -*Identity provider* in ABAC terminology- given the user's consent. The process can be automated since all data is electronically exchanged, and verification will be effected with the use of electronically signed credentials. Furthermore, clients requesting a loan may retain their anonymity if the banking institution decides that some non-revealing information from a reputable authority is sufficient to approve the client's request, for example instead of a full name and address the client presents only a Citizen's Identification Number from the Government Taxation Office.

4.1.1 Before ABAC

The current situation in the Greek Banking environment involves mostly manual procedures. The steps taken, starting from the request submission until the purchase of a banking product, may differ depending on the product value and the risk involved. Extensive screening and detailed verification of personal and financial data is applied for high risk products such as business loans, mortgages etc. Simpler procedures are followed for lower risk products like consumer loans and credit cards. In all cases however, a lot of paperwork is required with reference to the documents that must be submitted with the client request, and a considerable amount of time will be spent for the verification of the submitted data.

Although credit cards request-forms can be submitted by post, for all other products the client will have to eventually visit the bank.

A bank employee will create a new (potential) client record and receive the client's request-form along with the necessary documents that list the client's personal information (financial, social security, health insurance, etc.). The documents' authenticity will have to be verified either by simple inspection or, depending on the value of the banking product, by direct contact of the originating authority.

In the case of collaterals such as real estate, external partners such as real estate professionals may be called upon by the bank to visit and inspect the property on location.

In order to complete the client’s risk profile, the banking institution will collect additional data from organizations such as the National Banking Information System. This information concerns existing loans that the client has taken and loan payments that are overdue.

Finally, the banking institution may also collect information regarding the credit history of the client, and then calculate the risk of the client not being able to make the payments for the requested bank product.

At the end of the process the client’s request will be approved or rejected based on the risk score and the customer’s profile according to the banking institution’s policy.

The “loan origination” use-case closely adheres to the “attribute based access control” paradigm, considering that the user’s “banking profile details” are the attributes based on which the banking institution will control the user’s access to the institution’s services i.e. credit card issuance, consumer loan etc. The ABAC architecture provides considerable improvements regarding automation, efficiency, security, and privacy which are issues causing significant concern especially in the context of today’s banking environment.

4.1.2 After ABAC

Once the ABAC architecture is adopted, it is expected that the certificate-issuing authorities as well as the banking institutions will deploy servers offering ABAC functionality.

4.1.2.1 Identity Providers

The certificate-issuing organizations are the Identity Providers. They store and certify aspects of a user’s identity such as full name, address, telephone number, social security status, health insurance status, annual income, financial obligations (outstanding balance), etc.

4.1.2.2 Service Providers

The banking institution is the Service Providers offering its products to potential customers.

4.1.2.3 Behavioral Authentication Authorities

Organizations such as mobile telephony providers can function as Behavioral Authentication Authorities (BAA) providing second level authentication at the request of an Online Service. A mobile telephony provider may compare a user’s current location with the recorded user’s usual whereabouts based on the cell-towers that the user’s mobile device usually connects to. If there is no match the authentication will fail.

4.1.2.4 LATCH

Users may also configure BAAs to lock the account that a user has with an Identity Provider. In case multiple consecutive behavioral authentication attempts fail, a BAA’s ABAC server will use the LATCH service to lock all the user’s accounts according to the user’s specifications.

Taking all the above into consideration, the financial scenario is shaped as follows.

4.1.2.5 The Financial scenario revisited

A user that wishes to have a credit card or consumer loan issued by a banking institution will navigate to the banking institution’s web portal using the web browser of a mobile device (smartphone, tablet) or desktop computer. The bank’s web portal is the front-end to its ABAC infrastructure.

The user will be requested to provide certain information that will need to be validated. Such information includes, full name, age, annual income, social security status, employment status etc.

The user already has electronic credentials for some of these details stored in the mobile device. The user uses biometric authentication (fingerprint, voice, etc.) to unlock these credentials and submit them to the bank’s web portal.

The user also uses the mobile device to visit the web portals of other authorities and have electronic credentials issued for any additional information that the bank’s portal requests. The user either has a separate user-login with each of these authorities or may use OpenID and OAuth authentication in order to be identified at the authority’s server and have the necessary credentials issued. All the credentials are electronically signed by the issuing organization in order to ensure their authenticity. The credentials only include/reveal the specific information that the banking institution requests and not the total of the user’s records maintained by the credential-issuing authority. The issued credentials are transferred to the user’s mobile device and the user submits them to the bank’s web portal.

The bank’s ABAC server may in addition request for second level behavioral authentication. The BAA’s, e.g. Mobile Telephony Provider, ABAC server will retrieve the records of the user’s registered behavior, e.g. mobile cell-towers that the user mobile phone usually connects to, as an indication of the user’s normal whereabouts, and calculate the probability that the current user’s behavior matches the user’s profile or not. If the authentication fails, the BAA may also lock the user’s accounts at other ID Providers.

The bank’s ABAC server will examine and verify the credentials, consider the BAA’s response, and apply the banking institution “access control” policy. This policy specifies the criteria that must be fulfilled in order to accept or reject a client’s request for a credit card or consumer loan. If all criteria are met the user will be issued the credit card or granted the consumer loan.

4.2 Campus Wi-Fi and Campus-Restricted Web Services

The Campus Wi-Fi pilot focuses on the access of students, professors and guests to the Campus Wi-Fi and other Campus Web Services. The students and professors requesting access to one or more Campus resources (e.g. research network with the permission to print), are required to present information including personal (such as first and last name, age, street address, etc.) and educational (such as year of study, scholarship, teaching years, etc.) attributes. The guests must be vouched by a registered user in order to get access to some resources.

With the current infrastructure, the users reveal their full profile to get access to some resources with a lot of unnecessary information. They use the traditional username/password scheme to authenticate which nowadays is considered an insecure way. They also get access to various resources by simple access control policies.

The authentication and the authorization procedure can be vastly improved using the ABAC architecture. The pilot before and after ABAC and the advantages of using the ABAC architecture are presented below.

4.2.1 Before ABAC

When the students and professors register with their university, the university’s IT services create and store their full profile within their infrastructure.

A user that wishes to get access to the campus Wi-Fi and the other resources can navigate to the University’s web portal using the web browser of his/her mobile device.

Then the user uses his/her university username and password in order to authenticate to a particular web service against the university’s Authentication and Authorization Server. This centralized component has a complete view of the user’s profile and thus can provide the appropriate access to the resources.

The Authentication/Authorization Server maintains a role-based access control list. This has the disadvantage to not allow the definition of flexible fine-grained access control policies.

In summary, this paradigm has two important drawbacks; a user must reveal all of his identity to the Authentication/Authorization server and the network administrator is unable to define flexible fine grained access control policies. This results in privacy concerns from the user’s perspective.

4.2.2 After ABAC

When the students and professors register with their university, the university’s IT services create and store some identity attributes (such as full name, title or department, etc.) within their infrastructure. Furthermore, credentials of the identity attributes are issued and stored on user’s device.

A user that wishes to get access to the campus Wi-Fi and the other resources (with some permission) can open the Campus Access mobile application and choose the network resources that he wants to get access to.

Then the Campus Access mobile application informs him/her for the identity attributes required for all the selected resources.

After the user’s explicit consent to release the identity attributes, the user authenticates with the Authentication Server by using biometric authentication (e.g., fingerprint). This is achieved by utilizing FIDO credentials that were created during the user’s registration.

After the successful authentication, the Authentication Server transfers the required identity attributes to the Authorization Server via the OpenID Connect specification. To achieve this, the Authentication Server is configured as an OpenID Connect Provider and it maintains an attributes database which maintains the users’ identity attributes.

With the confirmed identity attributes, the Authorization Server checks against to its ABAC XACML-compliant access control policies and grants or denies access to those resources. The ABAC policies specify the criteria that must be fulfilled in order to accept or deny a user’s request. If all criteria are met the user can get access to the required resources.

4.2.2.1 Advantages of ABAC

1. The users have the opportunity to use a very useful mobile application to get access to various resources instead of the navigation to the University’s web portal to find a resource.

2. The users do not have to validate their complete identity and their full profile in order to get access to a resource. This approach allows users to reveal only a set of required identity attributes and not to reveal irrelevant parts of their profile.
3. The Authorization Server grants access to the requested resources according to the complex attribute-based access control policies instead of the simple policies. The ABAC policies also take into account the level of assurance of the required identity attributes and the level of criticality of the requested resources

4.2.2.2 Attributes and Attribute Values

The tables below present a list of identity attributes with a valid attribute value for a professor and a student of the Campus Wi-Fi and Campus-Restricted Web services use case scenario.

4.2.2.2.1 Professor:

Attribute	Attribute Values
First Name	John
Last Name	Andreou
Father's Name	Andreas
Gender	Male
Birth date	1970-02-15
Age	46
Nationality	Cypriot
Street Address	15, Athinon
Country	Cyprus
City	Nicosia
Postal Code	2710
E-mail	john.andreou@cut.ac.cy (university) , jandreou@gmail.com (home)
Phone Number	99-965423(mobile), 22-346529(home), 25-233303 (work)
Title	Professor
Department	Electrical Engineer, Computer Engineer and Informatics
Start Date	2009-09-10
End Date	-
Year of Study	-
Semester	-
Teaching Years	7
Chosen Courses	-
Teaching Courses	Control Systems, Electronics II, VLSI Systems Design
Number of passed courses	-
Scholarship	-

4.2.2.2.2 Student:

Attribute	Attribute Values
First Name	Helen
Last Name	Panayi
Father's Name	Stelios
Gender	Female
Birthday	1994-02-15
Age	22
Cypriot	Greek
Street Address	2, Machiton Eldyk
Country	Cyprus
City	Limassol
Postal Code	4651
E-mail	helen.panayi@cut.ac.cy (university) , hpanayi@gmail.com (home)
Phone Number	96-972325(mobile), 25-712093(home)
Title	Student
Department	Multimedia and Graphic Arts
Start Date	2012-09-10
End Date	2016-05-10
Year of Study	4
Semester	7
Teaching Years	-
Chosen Courses	Virtual Reality, Web Design, Operating Systems, Informatics
Teaching Courses	-
Number of passed courses	23
Scholarship	Yes

4.2.2.3 Resources (Authorization Server)

The table below presents a list of the resources for the Campus Wi-Fi and Campus-Restricted Web services use case scenario.

Resource ID	Resource (permission)
1	Internet
2	Moodle

3	Apps Web Resource (i.e., CUT Apps)
4	Webmail
5	Print
6	Research Network
7	File Server

4.2.2.4 Attribute-Based Access Control Policies

Some examples of the attribute-based access control policies of the Campus Wi-Fi and Campus-Restricted Web services use case scenario are presented below. Those examples demonstrate the flexibility that such an approach can offer.

“If a user is a **professor**”, he can get access to the **Internet**.

“If the user’s **first name** is **John** and his **last name** is **Andreou**”, he can get access to the **Internet**.

“If a user is a **professor**, belongs to the **Electrical Engineer, Computer Engineer and Informatics Department** and he is responsible for **teaching the Pattern Recognition course**”, he can get access to the **Print** resource.

“If a user is a **student**, belongs to the **Communication and Internet Studies Department**, he is in the **seventh semester** of his study and he **chose the Web Design course**”, he can get access to the **Research Network**.

“If a user is a **student**, belongs to the **Multimedia and Graphics Arts Department**, the **number of courses who has passed** is **more than 20** and his **current year of study** is **less than 6**”, he can get access to the **Webmail**.

“If a user has a **scholarship** and she belongs to the **Electrical Engineer, Computer Engineer and Informatics Arts Department**”, she can get access to the **Research Network** and **Print** resources.

“If a user is a **professor** with **more than two teaching years** in the CUT university”, he can get access to the **Research Network**.

4.2.3 Towards Privacy-Preserving ABAC

In order to be able to provide unlinkability and untraceability of end-users we will employ cryptographic credentials (Idemix and U-Prove) in the Wi-Fi pilot architecture. This will be achieved by making the appropriate changes to the mobile app so it can accommodate cryptographic credentials (and store them safely by utilizing TEE) and by enhancing the Authentication’s Server software stack. Specifically, we will make the below changes:

- **Mobile App:** The mobile app will be responsible for the secure storage of the cryptographic credentials. Also, it will encompass the Idemix and U-Prove stacks that will be responsible for releasing cryptographic credentials from the user device.
- **Authentication Server:** The Authentication will be responsible for the issuance and verification of cryptographic credentials. In a typical scenario, a user will visit the Authorization Server and then he will be redirected to the Authentication Server in order to verify some identity attributes. Subsequently, the user will present its cryptographic credentials to the Authentication Server, which will be responsible for verifying the credentials using the underlying Idemix or U-Prove stacks. After verifying the credentials, the Authentication Server will provide the verified identity attributes to the Authorization Server via the OpenID Connect specification.

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

4.3 Age Verification

Age verification is based on UPCOM’s Age Gate product, an attribute-based solution which can verify if a user requesting access to an age-restricted online resource is above a certain age, without revealing any other personal data. The online resource could be an age-restricted web site (e.g. porn or violence related), specific content (e.g. an NC-17 movie) or a purchase (e.g. alcohol or tobacco). The providers of those resources do not need to know any personal information of the users other than their age.

The Age Gate product uses various alternative methods in order to verify the user’s birthdate, such as trusted physical ID providers (government authorities, banks, universities, etc.) and electronic ID cards. It can then use ABAC in order to issue cryptographic credentials, including only the (verified) birthdate attribute, according to which the service provider can grant or deny access to specific resources, based on well-defined policies. Those credentials are issued to the user’s device, either by the ID consolidator or directly by a trusted IDP. The user can also backup those credentials to the IDC, so that they can be recovered, e.g. in case of device loss.

During the age verification application scenario, the following roles are identified:

- The user (user device), who acts both as a recipient, requesting age-related cryptographic credentials, and as a prover, against the online service that provides the requested age-restricted resource.
- The online service, who acts as the verifier of the user’s age.
- The ID Consolidator and the trusted physical ID Providers, who act as issuers, issuing age-related cryptographic credentials to the user’s device.

Following is the course of actions for the Age Verification scenario:

4. The user requests access to an age-restricted online resource.
5. The provider’s server requests from the Age Gate service the initiation of the age verification process.
6. The Age Gate creates and returns a QR code, which is displayed to the provider’s website.
7. The user scans the displayed QR code using the Age Gate mobile app.
8. The Age Gate mobile app authenticates the user (e.g. using biometric authentication).
9. Age Gate verifies the user’s age against the service provider’s policies, in order to grant or deny access to the requested resource.

4.3.1 Before ABAC

Age verification is an extremely important, yet very difficult to solve problem, especially without sacrificing user privacy. Current approaches fall into three main categories:

1. The user has to demonstrate evidence of ownership of a document which proves that he is an adult, such as a credit card or a driver’s license. However, there are major issues with this approach. First of all, sensitive information is inevitably revealed and personal data could also be disclosed. In addition, this approach cannot guarantee that the user is the legitimate owner of the provided proof. For example, a child could use her parent’s credit card or driver’s license, in order to access age-restricted content. Last but not least, proper age verification goes beyond the proof of being an adult. For example, the minimum age for drinking in USA is 21. Therefore, even if the user is the legitimate owner of a credit card, it does not necessarily mean that he is also above 21 years old.
2. The user has to demonstrate evidence of ownership of a document that explicitly states her birthdate, such as an ID card or a passport. With this approach, the service providers can determine the exact age of their users, however the rest of the problems with the first approach also apply here. Especially when it comes to the user’s privacy, such legal documents usually include even more personal data than a credit card.
3. The service provider includes an age disclaimer, with which the user must agree in order to gain access to an age-restricted resource. This is a widely-used approach, which is mainly used as legal cover for the service providers, hardly preventing any minors from accessing age-restricted resources.

4.3.2 After ABAC

We strongly believe that ABAC, along with the acquisition of the user’s physical identity, can offer a new approach to age verification, which provides a solution to all the problems related with the current approaches.

The exploitation of the ABAC architecture benefits all involved parties. More specifically:

- The users can access age-restricted resources without having to share with the service providers any personal or sensitive data other than the absolutely necessary (their birthdate).
- The service providers can verify the age of their visitors and grant them access to age-restricted resources, according to specific policies that they can easily create and manage.
- Minors are protected against age-restricted content and/or products.

4.4 ISIC Student Discounts

The ISIC Student Discounts pilot focusses on enabling service providers to create and present users with discounts based on specific user attributes. Discounts can be tailored to require specific attributes to allow the user to access and redeem a discount. In other words, a user will not be presented or be eligible for a discount if they do not meet the required attribute criteria put forth by the service provider. Verification of attributes by a service provider will be with the help of trusted IdP’s as opposed to relying on user for justification but will still be dependent on user consent to allow IdP’s to share any information. The user therefore retains control and insight into what is being shared with whom.

4.4.1 Before ABAC

Currently service providers collect attributes from user by consent to be able to target offers and discount to different segments of their customers. This results in multiple service providers collecting personal information about users without the user having any control or insight into how their attributes are used by the service provider after giving the initial consent.

Service providers are furthermore limited in tailoring discounts to customer segments because do not have access to a diverse set of user attributes and are restricted in acquiring this information from trusted IdP’s. Any personal information that is collected from a user is often rich data (i.e. proof of enrolment at a university) which may be disproportionate to the eligibility criteria for access to a discount (i.e. user is a student).

4.4.1.1 *Summary of the issues without ABAC*

The usage of ABAC capabilities in the ISIC Student Discount pilot will help to mitigate and solve the following issues:

1. Multiple service providers storing personal user information
2. User doesn’t have insight and control over their personal information
3. Service providers have access to rich data when assessing the eligibility of a user to access a service
4. Limited to no option to verify the user’s attributes through trusted IdP’s

4.4.2 After ABAC

In this pilot, the user is provisioned with a consent management in the Identity Consolidator, where the attributes can be managed. Also, the user can decide which attribute to be shared with which service provider.

The transaction based revenue model is used in this pilot. This means, percentages of the transaction amount are distributed to Issuer, Merchant and Student user. In this context the privacy of the user is maintained by only using three attributes, Transaction ID, Date and amount and only billing admin role is authorized to see this information. However, Merchants, Students, Issuers can see their own overview of the transactions.

When the student user performing a transaction, he can show the QR code based Student ID to the merchant and merchant can scan that QR code to verify the student status. Here the merchant sends a dynamic query to the Identity Provider to check the status, but doesn't store any attributes locally.

During the transaction process, if the service provider needs to verify the age of the user, that can be done in a private way, without knowing the date of birth of the user. The IdP can confirm with a binary response about the user age. This information is used by the service provider to qualify the user to access the service. For more complex attribute verification, multiple IdP's can confirm different criteria required for the eligibility of the user.

Overall, in student pilot, the privacy of the user is taken into account during registration, authentication, attribute verification and also included in the business processes.

5 Privacy and Security Considerations

5.1 Attacks and Privacy Issues in ABE and ABAC

In this section, we describe identified attacks and drawbacks from a privacy point of view for ABE and ABAC systems. In general, from our survey we have observed that although many ABE and ABAC schemes have been proposed in the literature, there is still no scheme that can eliminate all security and privacy loopholes found in these systems.

5.2 Lack of Revocation

User revocation is a major issue in ABE systems, where each attribute is conceivably shared by multiple users. As quoted in [52]: “Revocation of any single user would affect others who share his attributes. Moreover, user revocation in attribute based systems may be flexible and occur in different granularities. That is, it may require to revoke either the entire user access privilege, or just partial access right of the user, i.e., a subset of his/her attributes. Existing CP-ABE schemes suggest associating expiration time attributes to user secret keys.” For the same reasons described above, revocation is also difficult to be achieved in ABAC systems.

5.3 Key Abuse Attack for KP-ABE

This attack was presented in [53], where the key abuse attack is introduced. In the KP-ABE, a user secret key is defined over an access structure and does not have the one-to-one correspondence with any particular user. As a result, a paid user is able to “share” his secret key and abuse his access privilege without being identified. More seriously, malicious users may take this advantage to make profits by abusing the access privilege. We call this kind of misbehavior as key abuse attacks.

5.4 Key Escrow

Most of the existing ABE schemes are constructed on the architecture where a single trusted authority, or a key generation center has the power to generate the whole private keys of users with its master secret information. A major drawback with these schemes is known as a key escrow problem [54]. The key generation center could decrypt any kind of messages addressed to specific users by generating their private keys. This is not suitable for data sharing typical scenarios where the data owner would like to make their private data only accessible to designated users.

5.5 Attribute Hiding Attack in ABAC

ABAC policy may grant access to protected resources based on two complementary ways. At the first one the user should hold a set of attributes in order to gain access. At the second the user should not have the specific attributes (e.g., User should not be from Europe). In such a case ABAC is vulnerable to attribute hiding attacks [55]. In this attack, a malicious user hides or alternates some attribute values that he holds in order to get access to a resource that would be denied otherwise.

5.6 Revelation of Access Policy and Attributes to Untrusted Servers

In existing constructions of ABE, either the access policy or attributes should be attached in plaintext to the data ciphertext to facilitate user decryption. These plaintexts, particularly data access structures

in ABE, reveal the data owner’s access policies when disclosed to untrusted servers, and hence have privacy concerns.

5.7 Revelation of User's Identity in Multi Authority Scheme

This privacy issue of the CP-ABE scheme has been described (and resolved) in [56]. As the authors quote: “In multi authority attribute-based encryption schemes, a user can acquire secret keys from multiple authorities with them knowing his/her attributes and furthermore, a central authority is required. Notably, a user's identity information can be extracted from his/her some sensitive attributes. Hence, existing ABE schemes cannot fully protect users' privacy as multiple authorities can collaborate to identify a user by collecting and analyzing his attributes. Also the central authority has the power to decrypt every ciphertext, which seems somehow contradictory to the original goal of distributing control over many potentially untrusted authorities”.

5.8 Mitigations to Enhance Security in ABE and ABAC systems

In the literature, many ABE schemes have been proposed that try to eliminate the above security and privacy issues and at the same time try not to affect the performance of ABE systems. For example, in [57] the authors try to create a revocation scheme for ABE. In particular, the authors can revoke one attribute of a user instead of all attributes issued to him and the user can complete decryption as long as the unrevoked attributes of the user satisfy the access structure. A comprehensive review of various ABE schemes that address the identified issues for ABE can be found in [58] and [59].

5.9 Privacy considerations of Idemix and U-Prove

In this section, we analyze the privacy features and possible drawbacks of Idemix and U-Prove anonymous credentials. Despite the fact that these two systems provide significant privacy enhancements, we have observed that the literature does not include many works regarding their privacy issues or potential drawbacks.

5.9.1 Threat Model

Privacy properties are defined with respect to assumptions about adversaries, which could be inside the system, such as adversarial identity providers, or they could be external parties with access to some or all of the information available in the system. Adversaries can try to disclose information about user attributes or past actions through any available means. A party A that trusts a party B will believe not only that B is trustworthy, but also that B takes measures to prevent adversaries from gaining access to privileged and sensitive information, including mechanisms to defend against code vulnerabilities. Also, trust is not necessarily permanent. That is, party A might cease to trust B if evidence emerges that indicates B might be untrustworthy [60].

5.9.2 Comparison of Privacy features

Idemix features two kinds of credentials: Idemix pseudonyms and Idemix anonymous credentials. An Idemix pseudonym is used for two-party returning-user authentication. An Idemix anonymous credential, on the other hand, is used for third-party open-loop authentication, and provides full privacy, including unobservability, anonymity, selective disclosure (including the ability to prove that a numeric attribute is greater or less than a numeric constant without disclosing its value), issue-show unlinkability, and strong multi-show unlinkability by the same party or different parties. On the other hand, U-Prove has U-Prove tokens, which are used for third-party open-loop authentication. They

provide unobservability, anonymity, selective disclosure (but not the ability to prove that a numeric attribute is greater or less than a numeric constant without disclosing its value), and issue-show unlinkability. However, they do not provide multi-show unlinkability [61].

In U-Prove technology, the issuance of credentials is based on blind signatures. The issuer thus cannot link an issued credential to the issuing session. However, for proving validity or attributes of a credential, besides a zero-knowledge proof of knowledge of the recorded attributes, the issuer's signature is revealed. Different usages of the same credential are thus linkable. In order to ensure unlinkability, a used credential would have to be reissued after every usage. On the other hand, Idemix technology does not require revealing the signature for proving properties. Possession of a signature is proved with a zero-knowledge proof. Therefore, multiple shows of the same credential can remain unlinkable. As issuance is not based on blind signatures, the issuer can link an issued credential to its issuance session. However, this does not immediately reveal the link with subsequent sessions where the credential is shown. An important feature of Idemix technology is that it also allows to prove only properties of recorded attributes, such as a range of an attribute value. Further, it is possible to prove that committed values or values enclosed in a verifiable encryption are credential attributes [62].

5.9.3 Revocation

As mentioned in [63], unlinkability makes revocation difficult. The ability to revoke credentials is usually taken for granted. In the case of privacy-friendly credentials, however, it is difficult to achieve. As discussed in section 2, an ordinary CRL (Certificate Revocation List) cannot be used, since it would require some kind of credential identifier known to both the issuer and the relying parties, which would defeat unlinkability.

U-Prove credentials can be revoked by users because they do not have multi-show unlinkability, but cannot be revoked by issuers, because they have issue-show unlinkability. Idemix credentials, which have both multi-show unlinkability and issue-show unlinkability, are revocable neither by users nor by issuers. U-Prove credentials have a Token Identifier, which is a hash of the public key and the signature. Because U-Prove does not provide multi-show unlinkability, the Token Identifier, like the public key and the signature, is known to all the relying parties. The user agent could therefore revoke the credential by including the Token Identifier in a CRL. However, because U-Prove provides issue-show unlinkability, the credential issuer does not know the Token Identifier, nor the public key or the signature, and therefore cannot use it to revoke the credential.

Idemix has a credential update feature that can be used to extend the validity period of a credential that has expired. This facilitates the use of short-term credentials that may not need to be revoked. However, the credential-update feature can be used to implement credential revocation. Moreover, short term credentials are an alternative to revocation, but they have drawbacks:

- short term credentials are costly to implement for the issuer;
- they impose a logistic burden on the user agent;
- they may become unavailable if the issuer is down when the validity period needs to be extended;
- the user agent may be overwhelmed by the need to renew many credentials at once if it has not been operational for an extended period of time.

5.10 OpenAM P-ABAC Security Considerations

This section reports Security Considerations on the OpenAM-P-ABAC integration, described in Section 3.1.8.

The privacy of the end user is by maintaining anonymity in the authentication process. Especially, with the OpenAM and P-ABAC integration, the issuance of crypto credentials does not allow to identify the user. In the case where the user is accessing a new service, there is no traceability information stored in the ID consolidator, this will ensure there is unlinkability of the user activities. The platform itself do not monitor any activity of the user confirming the unobservability to maintain the privacy of the user.

For the security part, the components (OpenAM, FIWARE, etc.) are hosted in a secure environment. The Identity Providers, and the service providers are registered via secure communication. The access to the applications/platform is managed by establishing right controls such as higher levels of authentication.

5.11 FIDO Extensions for ABAC and Anonymous Credentials

This section reports Security Considerations on the FIDO-P-ABAC integration, described in Section 3.1.6.

FIDO aims at reliably identifying the user in order to authorize her. However, user identification may not always be needed or desirable, as it may jeopardize the user’s privacy. Our proposed extension to the FIDO UAF protocol, described in Chapter 6, aims at retaining the benefits of FIDO, especially in terms of usability, while employing anonymous credentials to implement attribute-based access control. Moreover, our proposed approach does not change substantially the FIDO UAF protocol, thus we do not foresee any additional security threats.

5.12 Security Considerations for the IRMA-FIWARE integration

In this subsection we refer to the integration between the IRMA implementation targeted at mobile devices and the FIWARE Privacy Open RESTful Specification presented in Section 2.2.2.1.

IRMA relies on JSON web tokens (JWTs) in its protocol steps. From the security point of view, the service provider must know the RSA key used by the IRMA API Server for JWT signatures and verify them consistently.

5.13 Privacy and Security Considerations in Idemix and U-Prove implementation

In U-Prove technology, the issuance of credentials is based on blind signatures. The issuer cannot link an issued credential to the issuing session. However, for proving validity or attributes of a credential, besides a zero-knowledge proof of knowledge of the recorded attributes, the issuer’s signature is revealed. Different usages of the same credential are thus linkable. In order to ensure unlinkability, a used credential would have to be reissued after every usage. Differently from U-Prove, Idemix provides the unlinkability capability by means of the usage of the concept of pseudonyms that allows the user to show the same credential for different sessions.

In terms of implementation, both the U-Prove protocol and Idemix protocol make use of RSA and ECC primitives which provide a level of security equivalent to a 128-bit symmetric key. Also, considering that these protocols involve the SHA-256 algorithm for hash functionalities, both protocols provide a satisfactory level of security in terms of resilience to brute-force/forging attacks. The privacy protection is given implicitly by the protocols specification. To eventually increase the level of security by generating private keys of 4096 bits from groups built starting from large numbers factorization problem or 512 bits from groups that are based on discrete logarithm problem, it is highly recommended to raise the hash algorithm used from SHA-256 to SHA-384 or even SHA-512.

The communication between the other components from ReCRED is performed through a TLS channel that ensures the confidentiality of the information and the authenticity of the servers. As an additional security measure, all the requests received by both the Idemix and U-Prove REST API must contain a session established together by issuer and prover at the beginning of the protocol. This ensures that once the protocol has started, no attacker can interfere with the protocol and alter the messages exchanged by the prover with the issuer.

Regarding the implementation of various sensitive components into the TEE we choose the private key generation and sign operation to be executed inside the trusted environment. This ensures that, once generated, the private key associated with the token or the master secret key associated to the user cannot be exported in an un-secure environment. Trustlet access is performed through a well-defined API and does not allow the disclosure of the private key/master secret key.

5.14 Security Analysis of the ABE-Based P-ABAC solution for Wi-Fi

This section reports Security Considerations on the ABE-Based P-ABAC solution for Wi-Fi, described in Section 3.1.11.

In our proposed system, the WLAN access point makes use of the WPA2 link-level encryption protocol, so that the access to the network requires knowledge of the pre-shared key (PSK). While common deployment of WLAN systems based on WPA2 remains vulnerable against password cracking attacks due to the weakness of the password/passphrase, we propose to generate a random key, of the maximum size allowed by WPA2 at each successful connection so that it cannot be easily guessed, at least assuming the usage of a secure RNG algorithm in the system. However, this challenge-like approach cannot be used without integration of an efficient technology for the distribution of the key that is not pre-shared, i.e., it is a priori unknown to the user. In order to make possible for the users to retrieve the secret key required to access the network by proving that they are authorized, we exploit the CP-ABE scheme. CP-ABE allows us to encrypt the PSK, defining a boolean access policy over the attributes that the user must own in order to be able to decrypt and retrieve the PSK. This strengthens the security of actual WPA2-based systems building an access control mechanism for key distribution on top of it. Assuming honest users that do not leak the CP-ABE secret key to other users with the aim of satisfying the policy, it will be impossible for an attacker to sniff or even forge packets to obtain access to the network without having the credentials required by the encryption policy. Indeed, the random generated WPA2 key will be encrypted by means of the CPABE scheme and decrypted only by the legitimate users who are able to satisfy the policy.

6 Conclusions / Future Work

This deliverable entitled “Full Design and Prototype of the ABAC Infrastructure” follows and improves the architecture defined in D5.1 “Specification and Initial Design of the ABAC Infrastructure”. This document provides the description of the ReCRED Privacy-Preserving Attribute-Based Access-Control (P-ABAC) infrastructure integrated in the ReCRED framework to provide ABAC capabilities with the guarantee of privacy protection of the user information. In this deliverable, the architectural design has been enhanced with the definition of revocation systems as well as the integration with commercial solutions for authentication like OpenID-connect and FIDO protocols in order to provide a platform compatible with already deployed commercial solutions. The design of actual P-ABAC modules has been supported by the final prototyping and the integration of such components to have the final prototype of the whole P-ABAC architecture to be integrated in the ReCRED framework (e.g. the Credential Management Module is already integrated in the Identity Consolidator module defined in WP4).

The results of this deliverable will be used as input for the integration activities and for the deployment of the pilots. Future work, includes the integration of other cryptographic schemes in the P-ABAC architecture. Indeed, the first implementation of ABE scheme in the WiFi use-case will drive the final integration of it in the final P-ABAC architecture in order to provide a full and flexible solution for credential systems.

7 References

- [1] Lapon, Jorn, et al. "Analysis of revocation strategies for anonymous Idemix credentials." IFIP International Conference on Communications and Multimedia Security. Springer Berlin Heidelberg, 2011.
- [2] Camenisch, J.L., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003).
- [3] Backes, M., Camenisch, J., Sommer, D.: Anonymous yet accountable access control. In: Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, pp. 40–46. ACM, New York (2005).
- [4] Camenisch, J., Kohlweiss, M., Soriente, C.: Solving revocation with efficient update of anonymous credentials. In: Security and Cryptography for Networks, pp. 454–471 (2011).
- [5] Nakanishi, T., Fujii, H., Hira, Y., Funabiki, N.: Revocable group signature schemes with constant costs for signing and verifying. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 463–480. Springer, Heidelberg (2009)
- [6] Camenisch, J.L., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
- [7] Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)
- [8] Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (2009)
- [9] IRMA (I Reveal My Attributes) Project. <https://www.irmacard.org/>
- [10] IRMATube. <https://demo.irmacard.org/v2/demos/irmaTube/>
- [11] Advanced crypto software collection ciphertext-policy attribute-based encryption. <http://hms.isi.jhu.edu/acsc/cpabe/>.
- [12] Linux wireless - hostapd. <http://linuxwireless.org/en/users/Documentation/hostapd/>.
- [13] Linux wireless - iw. <http://linuxwireless.org/en/users/Documentation/iw/>.
- [14] Linux wireless - wpa supplicant. <http://linuxwireless.org/en/users/Documentation/wpa-supPLICANT/>.
- [15] IEEE standard for information technology telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - part 11: Mac and phy specifications. IEEE Std 802.11-2007, pages 1–1076, June 2007.
- [16] IEEE standard for local and metropolitan area networks—port-based network access control. IEEE Std 802.1X-2010, pages 1–205, Feb 2010.
- [17] Cisco visual networking index: Global mobile data traffic forecast update, 2015 white paper. <http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html>, 2016.
- [18] A. Aijaz, H. Aghvami, and M. Amani. A survey on mobile data offloading: technical and business perspectives. Wireless Communications, IEEE, 20(2):104–112, 2013.
- [19] W. Alliance. Wpa2 security now mandatory for Wi-Fi certified products. Press Release, 2006.
- [20] M. Ambrosin, M. Conti, and T. Dargahi. On the feasibility of attribute-based encryption on smartphone devices. In Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems, pages 49–54. ACM, 2015.

- [21] P. Arana. Benefits and vulnerabilities of Wi-Fi protected access 2 (wpa2). INFS 612, pages 1–6, 2006.
- [22] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin. Persona: an online social network with user-defined privacy. In ACM SIGCOMM Computer Communication Review, volume 39, pages 135–146. ACM, 2009.
- [23] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In Proceedings of the IEEE Symposium on Security and Privacy, SP’07, pages 321–334. IEEE, 2007.
- [24] H. Boland and H. Mousavi. Security issues of the ieee 802.11b wireless lan. In Proceedings of the Canadian Conference on Electrical and Computer Engineering, 2004., volume 1, pages 333–336. IEEE, 2004.
- [25] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. ACM SIGCOMM Computer Communication Review, 28(4):56–67, 1998.
- [26] A. Cassola, E.-O. Blass, and G. Noubir. Authenticating privately over public Wi-Fi hotspots. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS’15, pages 1346–1357. ACM, 2015.
- [27] A. Cassola, W. K. Robertson, E. Kirda, and G. Noubir. A practical, targeted, and stealthy attack against WPA enterprise authentication. In Proceedings of the 20th Annual Network and Distributed System Security Symposium, NDSS’13, 2013.
- [28] T. Dargahi, M. Ambrosin, M. Conti, and N. Asokan. Abaka: A novel attribute-based k-anonymous collaborative solution for lbss. Computer Communications, 85:1–13, 2016.
- [29] K. Friksen, M. Atallah, and J. Li. Attribute-based access control with hidden policies and hidden credentials. Computers, IEEE Transactions on, 55(10):1259–1270, 2006.
- [30] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Proceedings of the 13th ACM conference on Computer and communications security, CCS’06, pages 89–98. Acm, 2006.
- [31] V. C. Hu, D. Ferraiolo, R. Kuhn, A. R. Friedman, A. J. Lang, M. M. Cogdell, A. Schnitzer, K. Sandlin, R. Miller, K. Scarfone, et al. Guide to attribute based access control (ABAC) definition and considerations. NIST Special Publication, 800:162, 2013.
- [32] V. C. Hu, D. R. Kuhn, and D. F. Ferraiolo. Attribute-based access control. Computer, (2):85–88, 2015.
- [33] T. Jung, X.-Y. Li, Z. Wan, and M. Wan. Control cloud data access privilege and anonymity with fully anonymous attribute-based encryption. IEEE Transactions on Information Forensics and Security, 10(1):190–199, 2015.
- [34] J. S. Park and D. Dicoi. Wlan security: current and future. IEEE Internet Computing, 7(5):60, 2003.
- [35] A. Sahai and B. Waters. Fuzzy identity-based encryption. In Advances in Cryptology–EUROCRYPT 2005, pages 457–473. Springer, 2005.
- [36] S.-Y. Tan and W.-S. Yap. Cryptanalysis of a CP-ABE scheme with policy in normal forms. Information Processing Letters, 116(7):492–495, 2016.
- [37] H. Zhang, X. Chu, W. Guo, and S. Wang. Coexistence of Wi-Fi and heterogeneous small cell networks sharing unlicensed spectrum. Communications Magazine, IEEE, 53(3):158–164, 2015.
- [38] FIDO Alliance – <http://fidoalliance.org>
- [39] OpenID – <http://openid.net>
- [40] OpenAuth – <http://oauth.net>

- [41] Camenisch, Jan. "Specification of the Identity Mixer Cryptographic Library, Version 2.3.1, December 7, 2010."
- [42] Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In *Advances in Cryptology – EUROCRYPT 2001*, vol. 2045 of LNCS, pp. 93-118. Springer Verlag, 2001.
- [43] Camenisch, Jan, and Van Herreweghen, Els. "Design and implementation of the idemix anonymous credential system." *Proceedings of 9th ACM Conference on Computer and Communications Security*. ACM Press, 2002
- [44] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *Security in Communication Networks*, Third International Conference, SCN 2002, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer Verlag, 2003.
- [45] U-Prove Cryptographic Specification V1.1 Revision 3 - Christian Paquin, Greg Zaverucha
- [46] Chase, Melissa. "Multi-authority attribute based encryption." *Theory of cryptography*. Springer Berlin Heidelberg, 2007. 515-534.
- [47] Lewko, Allison, and Brent Waters. "Decentralizing attribute-based encryption." *Advances in Cryptology–EUROCRYPT 2011*. Springer Berlin Heidelberg, 2011. 568-588.
- [48] FIWARE - <https://www.fiware.org/>
- [49] FIWARE Privacy Open RESTful API Specification
https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Privacy_Open_RESTful_API_Specification
- [50] ABC4Trust - <https://abc4trust.eu/>
- [51] WS-Trust 1.4 - <http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/ws-trust.html>
- [52] Shucheng Yu, Cong Wang, Kui Ren, Wenjing Lou, "Attribute based data sharing with attribute revocation", *ASIACCS 2010*: 261-270
- [53] Shucheng Yu, Kui Ren, Wenjing Lou, Jin Li, "Defending Against Key Abuse Attacks in KP-ABE Enabled Broadcast Systems", *IACR Cryptology ePrint Archive 2009*: 295 (2009)
- [54] Xing Zhang, Cancan Jin, Zilong Wen, Qingni Shen, Yuejian Fang, Zhonghai Wu, "Attribute-Based Encryption Without Key Escrow", *ICCCS 2015*: 74-87
- [55] Jason Crampton, Charles Morisset, "PTaCL: A Language for Attribute-Based Access Control in Open Systems", *POST 2012*: 390-409
- [56] Jinguang Han, Willy Susilo, Yi Mu, Jianying Zhou and Man Ho Au, "Improving Privacy and Security in Decentralized Ciphertext-Policy Attribute-based Encryption", *IEEE Transactions on Information Forensics and Security*, Vol. 10, Issue 3, pp. 665 - 678, 2015
- [57] Qiang Li, Dengguo Feng, Liwu Zhang, "An attribute based encryption scheme with fine-grained attribute revocation", *GLOBECOM 2012*: 885-890
- [58] Zhi Qiao, Shuwen Liang, Spencer Davis, Hai Jiang, "Survey of attribute based encryption", *SNPD 2014*: 1-6
- [59] Cheng-Chi Lee, Pei-Shan Chung, Min-Shiang Hwang, "A Survey on Attribute-based Encryption Schemes of Access Control in Cloud Environments", *I. J. Network Security* 15(4): 231-240 (2013)
- [60] Eleanor Birrell, Fred B. Schneider, "Federated Identity Management Systems: A Privacy-Based Characterization", *IEEE Security & Privacy* 11(5): 36-48 (2013)
- [61] Francisco Corella, Karen Lewison, "Privacy Postures of Authentication Technologies", *ID360 Conference on identity*, 2013
- [62] Pros and Cons of Idemix for NSTIC – Pomcor, [Online] <https://pomcor.com/2011/10/10/pros-and-cons-of-idemix-for-nstic/>

- [63] Pros and Cons of U-Prove for NSTIC, [Online] <https://pomcor.com/2011/10/04/pros-and-cons-of-u-prove-for-nstic/>
- [64] Department of Defense Trusted Computer System Evaluation Criteria, DoD 5200.28-STD, December, 1985.
- [65] NIST Joint Task Force (2013). Security and privacy controls for federal information systems and organizations. NIST Special Publication 800, 53.
- [66] Ferraiolo, David, Janet Cugini, and D. Richard Kuhn. "Role-based access control (RBAC): Features and motivations." Proceedings of 11th annual computer security application conference. 1995.
- [67] Sandhu, R., Ferraiolo, D.F. and Kuhn, D.R. (July 2000). "The NIST Model for Role Based Access Control: Toward a Unified Standard" (PDF). 5th ACM Workshop Role-Based Access Control. pp. 47–63.
- [68] Ahn, Gail-Joon, and Ravi Sandhu. "Role-based authorization constraints specification." ACM Transactions on Information and System Security (TISSEC) 3.4 (2000): 207-226.
- [69] Claudio Pisa, Alberto Caponi, Tooska Dargahi, Giuseppe Bianchi, Nicola Blefari-Melazzi. "WI-FAB: attribute-based WLAN access control, without pre-shared keys and backend infrastructures". Proceedings of the 8th ACM International Workshop on Hot Topics in Planet-scale mObile computing and online Social networking (HotPOST) 2016.