University of Piraeus
School of Information Technology and Communications
Department of Digital Systems
Postgraduate Programme: Digital System Security

Master Thesis Title:

# Internet anonymity using anonymous credentials

Author:
Aristeidis E. Farao

Student ID no:
MTE1634

Supervisor:
Associated Professor, Christos Xenakis

Submitted to the Department of Digital Systems
in partial fulfillment of the requirements for the degree of

Masters of Digital System Security
at the
University of Piraeus

Piraeus, February, 2018

# Contents

# List of Figures

# List of Tables

# Acknowledgments

First and foremost, I would like to thank my supervisor, Christos Xenakis, for his mentorship and guidance over the course of this research. His ideas were key to the completion of this research and thesis. Our conversations helped me not only formulate, solve and explore the problems presented here, but also develop a more sophisticated understanding of Internet Security and Anonymous Credentials. His contribution to this research, including his generous expenditure of time in meetings and in the writing and revision of this thesis, are greatly and appreciated by the author.

I would also like to thank all the group members in the Security Systems Laboratory on the Department of Digital Systems for their support and help during various point of this work and for the fruitful, enjoyable conversations about both research and other topics.

Finally, I would like to thank my parents. Without them, regardless of everything else, none of the work in this thesis would have been possible.

*Piraeus, February 2018*
*Aristeidis Farao*

# Chapter 1

# Abstract

The everyday use of electronic services has increased significantly. For the enforcement of these transactions customers have to manage many different accounts in different service providers. The consequence of this action, is the customer's personal data exposure which is inevitable. Taking this into account, we find out how important the protection of our privacy is, because with this trend our privacy is lost.

There is a solution to this big issue and is called Anonymous Credential. Anonymous Credential Systems promise that they can change this situation and provide privacy and strong authentication. Nevertheless, the biggest disadvantage of the Anonymous Credentials is that these are still under research and they are very complex and not so understandable to the readers who are not researchers or experts in the domain of cryptography and mathematics. In this master thesis, our key goal is to provide a way so that the Anonymous Credential become more understandable to more people than before. Also, we demonstrate a new project which has as key goal the support of Anonymous Credential to their customers for their online and other transactions. Furthermore, we represent two use cases which will explain in detail how the Anonymous Credential will be used in real-life.

**Keywords:***Anonymous Credential, Anonymous Credential Systems, Credential, Zero-Knowledge-Proof, Pseudonyms,Privacy, Anonymity, User, Issuer, Verifier, Service Provider, IdeMix, U-Prove, IRMA, ABC4Trust, ReCRED.*

# Chapter 2

# Introduction

Internet is a part of our everyday life, the online-transactions are the result of the modernization of the existing internet technology in such a way that customers can do whatever they want from their smart devices which are connected to the internet. However, the biggest disadvantage into this revolutionary idea which improves our daily life is the anonymity issue.

Nowadays everyone understands how important anonymity is. Everyone wants to use services which have got some limitations and every user must provide some personal data to become available to her. However, users fill in forms which ask for more information than necessary. For example, if a service has got as limitation only the user's birthday year for her access, then the provided form will ask information which is about the name, the surname, the sex and the birthday year of the user. These are useless information for the Service Provider (SP) and have not to be asked and be collected.

SP is everyone who provides an online-service or an offline-service with an access policy and customers must meet the access policy's requirements. These needed requirements from the user's side are called access policy, based on this policy a user will be accepted or rejected. From these illegal actions each SP can create a profile for these users and probably have profits from this action. This is something which a user does not desire because she may be exploited.

Traditionally, the access control has been based on the user's identity which is revealed to the SP and then he will choose if the user is able to use this service. This identity based access control system is not an anonymous way because the user must disclose much more than the necessary information to gain access into a service.

Nevertheless, there is a solution which is named Anonymous Credential (AC), this can provide help to the users to escape from this unwanted situation. AC is a credential which contains owner's information but discloses just what is needed to convince a SP that the credential's owner meets the service's requirements and is able to use the service. The ACs provide access control based on attributes which are called Attribute Based Access Control (ABAC) and is based on user's attributes and on service's access policy. The definition of ABAC is an access control method where the requests to perform operations on objects are granted or denied based on assigned attributes of the object, environmental conditions, and a set of policies that are specified in terms of those attributes and conditions.

AC can provide users with some special credentials which contain user's information and is able to provide privacy and strong authentication to the users, based on [1], [2] and [3]. This solution contains three (3) participants who are the user, the Issuing Organization (IO) and the SP. The AC is issued by an IO and is delivered to a SP. Also, this type of credential contains pseudonymous identity of the user. The real user's information is cryptographically stored and will never be revealed in their real way but only under a process which shall provide only the proof on some question. Moreover, the user can use the AC more than one-time without any chance of linking the user's actions.

Furthermore, the big advantage of the AC is that the SP cannot get the full credential with the data being revealed and cannot reuse it to impersonate the real user in one other SP. ACs provide anonymity, which means that someone else can see the user's actions but is not able to recognize who this user is. This is very important because in our life it is necessary to have freedom i.e. the anonymity can promote the free speech which is a legal human right.

Most of our daily transactions take place through Internet applications which are capable of controlling and tracing the actions without anonymity.

Based on the above facts we understand that the ACs can solve all the above problems and lead us to an anonymous community. So, there is a need for a thesis which will analyze how the AC works, why someone should prefer to use ACs and how these will be used in real life.

There are papers which demonstrate how the AC protocols work but these are basically for people who are experts in the cryptography and in mathematics. If this technology remains in the same situation never will it become approachable for other people who are not expert in these two scientific domains, but they have got insight into other different domains of computer science. If more people learn and understand the ACs then more applications will be created so the people will find at last what they want through these new applications which will provide anonymity.

In this thesis, we will present the importance of anonymity on the internet, solutions to achieve privacy. Also we will provide the meanings of Zero-Knowledge-Proof (ZKP), of pseudonyms and AC which are the base of the Anonymous Credential System (ACS). In the next chapters we will try to analyze in detail the AC, their components and their protocols. Also, we will demonstrate the current implementations and provide a use case based on the newest implementation.

The remaining parts of this Master Thesis are organized as follows. Chapter 3 demonstrates and explains the required theory-knowledge which the reader should obtain to understand the theory of AC. Chapter 4 presents the protocols of AC which are explained in detail. Chapter 5 introduces the ReCRED project and pays attention to AC's implementations. At last, the Chapter 6 displays two different use cases which will help the reader understand how the AC will be used in our real-life.

# Chapter 3

# Background

In this section we introduce the definitions of Zero-Knowledge-Proof, of Pseudonym System and we also provide the related work, the limitation and the current implementations of the ACS.

## 3.1 Zero-Knowledge-Proof

Zero-Knowledge-Proof (ZKP) was proposed by Goldwasser et al., [4] in February 1989 and it is a way to restrict the quantity of disclosed information which is transferred from a customer to a SP using a protocol. AC uses the ZKP to ensure the knowledge of a secret between the participants. Usually, when someone wants to convince and prove the possession of the knowledge of a piece of information, she must disclose the whole information. This action however, is quite risky, the verifier or a possible eavesdropper may use the disclosed information to carry masquerade or impersonation attacks. ZKP protocol belongs to interactive protocols, this means that the participants exchange a number of messages which are based on random numbers and anytime one of the participants can terminate the protocol. ZKP protocols are used for authentication and identification.

The idea behind ZKP is to prove the knowledge of a statement, without revealing any information about it. A statement is a requirement which the user meets or not, but must prove it to the verifier, i.e. a statement is that the user is over eighteen (18) years old. This secret information which is used is all legitimate and available only for a specific period, so every other period these information are totally useless. This secret is known only to the person who is asked to authenticate herself. This is achieved by the employed ZKP protocols of ZKP that allow such actions to be materialized. These are cryptographic protocols that do not disclose any secret information during their operation, (i.e., the secret information is never transferred to other parties), while the owner (i.e., the user) is able to prove that she knows these secrets, which are used for identification and/or authentication.

The prover tries to prove the knowledge of a secret (i.e., without revealing it) to a verifier, who asks questions about the secret information. From this question, the verifier can discover if the prover really knows the "secret" or tries to cheat on him. Only if the user knows the secret can she answer correctly to the provided questions and if she follows the rules. They set how the information will be transmitted and given by the system for the protection of the communication's members from an evil person who can be a third party or one of the conversation's members. The question is only one and is repeated to increase the level of confidence that the user really possesses the secret information and she does not try to cheat on the verifier.

A ZKP protocol should satisfy three main properties:

- **completeness** which means that if a statement is true, then the verifier will have been convinced of this fact by an honest prover

- **soundness** which means that if a statement is false, then an honest prover can convince the honest verifier that it is true. A very small probability of a failure exists regardless of the used implementation, the failure can be minimized if the repeated timed increased. Failure means that the prover managed to cheat on the verifier

- **zero-knowledge** which means that if a statement is true, an honest verifier learns nothing about the secret values of the statement but only the fact that the statements is true. The verifier can realize that the prover does not possess or know the required secret with a very high probability

Currently, there are many ZKP implementations, the most famous implementations are Fiat and Shamir [5], Uriel Feige, et al. [6], Jeans-Jacques

Quisquaterm et al. [7], Manuel Blum et al. [8], Schnorr [9], etc., which mainly follow the generic structure described below and depicted in Figure 3.1. The security of the most famous ZKP protocols depends on the discrete logarithm or in the factorization of a complex logarithm. A ZKP is executed between the prover and the verifier in four steps, which are enumerated below:

**Step 1:** At this step the protocol is initialized, and the prover sticks to some secret random integer numbers which will be used for the next message creation. Then the prover sends to the verifier a calculated value from the committed numbers.

**Step 2:** After receiving the committed message from step 1, the verifier randomly chooses a challenge, the value of challenge depends on the ZKP protocol which is used because each protocol uses a different way for the challenge's calculation, and sends it to the prover. The prover does not know in advance the verifier's challenge.

**Step 3:** At the third step the prover receives the verifier's challenge and calculates in polynomial time the response based on the verifier's challenge, which varies from the ZKP protocols which is used, and sends it back to him.

**Step 4:** The verifier checks the correctness of the response, with calculation which contains the values which come from the prover at the step 1 and step 3. This calculation depends on the ZKP protocol which is used because each protocol uses different values and information. These steps can occur as many times as needed to convince the verifier that the prover really knows the secret and does not try to cheat on him.

## 3.2 Pseudonym System

Each user who participates in an ACS owns one or more pseudonyms, that enable her to access anonymously to online services hiding her real identity from the SP or anyone else who monitors the Internet's traffic. For the generation of a pseudonym, the user generates a pair of keys that contain a master secret and a master public key which derives from the user's unique corresponding master secret key and these values come from a probabilis-

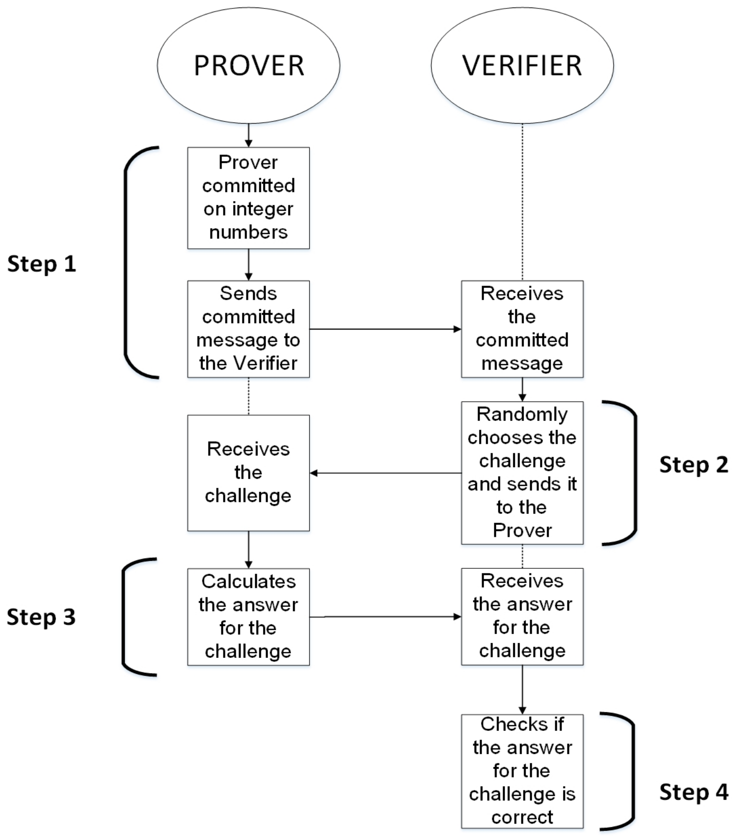

PROVER
VERIFIER
Prover committed on integer numbers
Step 1
Sends committed message to the Verifier
Receives the committed message
Receives the challenge
Randomly chooses the challenge and sends it to the Prover
Step 2
Step 3
Calculates the answer for the challenge
Receives the answer for the challenge
Checks if the answer for the challenge is correct
Step 4

Figure 3.1: Structure of zero-knowledge-proof

tic polynomial-time procedure, based on [1], [10]. Many pseudonyms can be generated by the same master secret key and its value is not chosen by the user, but this calculation is based on an interactive protocol which occurs between the user and the IO who will issue the user's pseudonym. The pseudonym is just a master public key which represents the user who owns it in every service. Nobody can understand if two pseudonyms come from the same secret master key. Furthermore, the user must keep it in a safe place and never disclose this master secret key because it is the locker that is able to reveal the user's real identity and deanonymize all its pseudonyms.

Moreover, a pseudonym can be used either once to access an online service or it can be used as many times its owner wishes to access different services, considering the privacy policy as well as usability issues (i.e., sometimes it is practical to use the same pseudonym for one specific service, so the user will not need to fix her options each time she uses the specific service). Each pseudonym is unique and cannot be re-generated again. This fact also safeguards both users and SPs from credential sharing or borrowing. For example, if Alice has got a pseudonym for a specific video club and her friend Bob wants to see a movie from this club but he does not own a credential for this club, then Alice cannot just share her pseudonym with Bob. She should share with him her secret master key which means that she will disclose to Bob all her ACs (i.e., all her pseudonymous) that will lead to very important consequences i.e. impersonation.

Furthermore, the pseudonyms are unlinkable, so two SPs cannot blend their data to create a folder for a specific user. A pseudonym can be used for the issuance of another pseudonym, so the user who will use the second pseudonym will not reveal the first pseudonym. In addition, the pseudonym model [10] consists of five (5) procedures that are executed between the user and the Issuing Organization (IO) who issues the ACs (i.e. the IO) and the SP as depicted in Figure 3.2. The steps for the pseudonym's creation are the following:

**Step 1:** Initially, all the participants, the user, the IO and the SP generate their master key pair, so they own a master secret key and a master public key.

**Step 2:** Then, the process "Registration with the Certification Authority" takes place in user's and IO's side where the user sends to the IO her master public key. With the master public key, the IO learns the user and it also ensures that the user is a valid user and then issues the credential for the specific user.

**Step 3:** Thirdly, the process which is called "Registration with an Organization" takes place between the IO and the user, where the IO and the user together will generate the user's pseudonym.

**Step 4:** Fourthly, the process which is called "Issue of Credential" takes place between the user and the IO, where the IO and the user will issue the credential based on a specific pseudonym based on an interactive protocol to issue the AC.

**Step 5:** At last, the process which is called "Transfer of Credential" takes place between the user and a SP, where the user who already owns a credential proves that she really owns a legal credential to any SP without revealing any extra information about herself. With the previous steps the user will become able to use the SP's service.

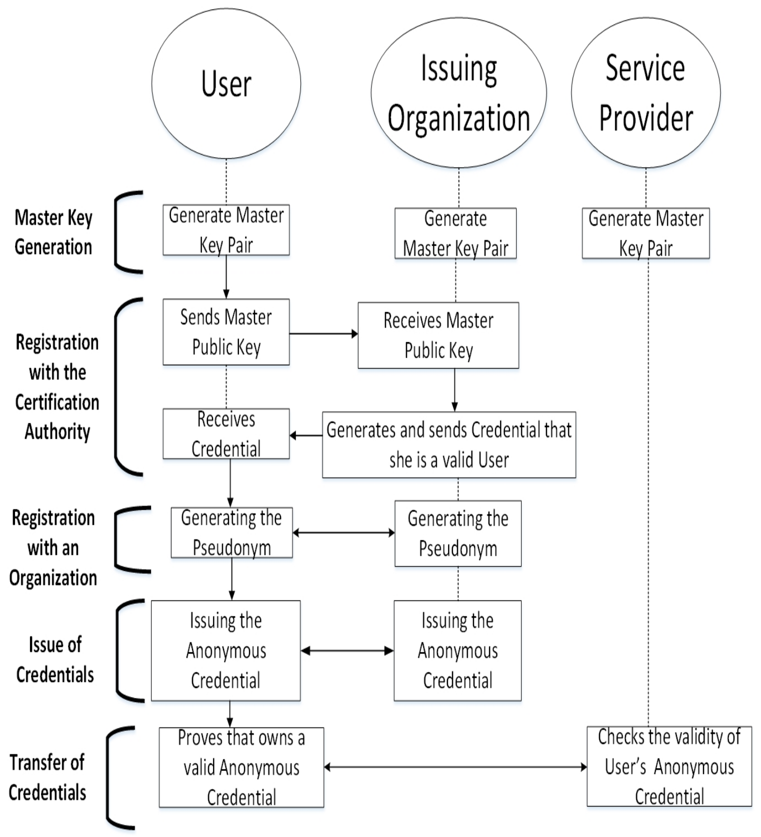

User
Issuing Organization
Service Provider
Master Key Generation
Generate Master Key Pair
Generate Master Key Pair
Generate Master Key Pair
Registration with the Certification Authority
Sends Master Public Key
Receives Master Public Key
Receives Credential
Generates and sends Credential that she is a valid User
Registration with an Organization
Generating the Pseudonym
Generating the Pseudonym
Issue of Credentials
Issuing the Anonymous Credential
Issuing the Anonymous Credential
Transfer of Credentials
Proves that owns a valid Anonymous Credential
Checks the validity of User's Anonymous Credential

Figure 3.2: Pseudonym creation and use

## 3.3 Related Work

Starting from the past '83 where D. Chaum in [11] proposed a way to achieve untraceable payments. This implementation could be used during an anonymous vote system and in payments. Moreover, in '85, when Sha Goldwasser, Silvio Micali and Charles Rackoff invented the ZKP in [4]. This was a revolutionary idea which would not reveal the attributes but only proofs about the user. We can easily understand that this is the basic and the most important skill of the ACS. Some years later, this technology went further by Uriel Feige, et al. in '87 where they proposed a ZKP protocol based on the factorization of a complex logarithm [6] and from the Schnorr's protocol which proposed another protocol based on the discrete logarithm, [9] in '89. Both two ideas were used in AC. However, Schnorr's protocol is used in IdeMix. With the ZKP the prover succeeds in convincing the verifier that the prover really knows the secret information.

ACs were invented by David Chaum in [11] in '83. Also, the pseudonym system was introduced by David Chaum in '85, [12] to interact anonymously with many organizations. In '99 Anna Lysyanskaya, et al. in [10] demonstrated the pseudonyms system, how important these are and how these will improve people's transactions in their life. In 2001 Jan Camenisch and Anna Lysyanskaya created their implementation which was the first implementation of the ACs [13]. In their paper, in "An Efficient for non-transferable Anonymous Credentials with optional Anonymity Revocation" in 2001, [13] they described the way that this idea could work with the protocols and how this is established. Furthermore, in this paper they described the participants and the role which each participant should follow to make this work perfectly. In addition, in the paper of Jan Camenisch and Els Van Herreweghen, in Design and Implementation of the IdeMix anonymous credential system in 2002, in [14], they described how the Identity Mixer works and gave an example scenario on how should be used in real life. In 2003 in [15] J. Camenisch and A. Lysyanskaya proposed a practical signature scheme and the corresponding protocols.

There are some implementations of ACSs. One of the most famous implementation is the implementation of IdeMix from IBM Research Zurich which, based on [13], [16] and [7], is an identity management system based on ACs and ZKP. IdeMix will be explained in detail in the following chapters. Also, there is the implementation of U-Prove in [30]. The implementation of U-Prove from Microsoft is based on public key cryptography, elliptic curves and hash functions. The U-Prove uses the U-Prove token which contains the attributes and cryptographically protects them against tampering. Moreover, there is the implementation of IRMA from Radboud university of Nijmegen, in [17]. IRMA is based on non-trivial cryptography for attribute-based credentials which are data that contain a provable claim, to achieve this IRMA uses both IdeMix and U-Prove. Furthermore, there is one other implementation which is called ReCRED. ReCRED binds the above technologies of IdeMix and U-Prove and some other technologies. ReCRED provides an authentication system, management of online user accounts and the issuance of ACs. Moreover, there is another implementation which is called ABC4Trust. The goal of ABC4Trust is to provide an implementation of attribute based credential systems where accredited members of communities will provide AC service to their members. To achieve this goal, ABC4Trust uses both IdeMix and U-Prove technologies.

## 3.4 Implementations

In this section we will represent the currently available implementations of AC. These implementations are the Identity Mixer, the U-Prove, the IRMA and the ABC4Trust, these will be analyzed in the following subsections.

### 3.4.1 Identity Mixer (IdeMix)

Identity Mixer is one of the most successful implementations of an ACS. This system was implemented by Jan Camenisch et. al., [20]. In this implementation there are three participants and two (2) protocols with some sub-protocols. The participants are the user, the issuer and the verifier. The user is everyone who will use this application and wants to be a part of this anonymous system, the issuer could be an organization or a bank which the user trusts and can issue the ACs and the verifier is each organization who provides a user with a service and wants some specific credentials to let the user to use the service. The two basic protocols which are used are:

- **Protocol** 1: “The Issuing Protocol” which takes place between the user and the IO where the user becomes a Recipient because she waits to issue a credential for her and

- **Protocol** 2: “The Show Protocol” which takes place between the user and the SP, where the user becomes a Prover because she will prove that she owns the right to use the Service.

We understand that the trust between the participants is a very important element for anonymity. Also, we can see that the trust between the user and the IO is necessary for their good cooperation. We can understand that the user must be sure that the IO will not reveal his personal data to anyone under any circumstances, because in opposite situation the anonymity will be lost and the system of ACs will be destroyed because after the reveal of these data the link between actions and the real people will be a very easy job. On the other hand, we have the IO and the SP who are very important participants. These roles sometimes can be played by different participants but also there are times when the SP and IO will be the same person. In case, these two are different there should be trust between them.

However, the trust between them is off-line because they are not bound by any protocol. The SP wants to be sure that the Credentials which it receives from each user are formed correctly and contain real user's information. The IdeMix implementation cannot only be used through the end user devices such as computers and mobile devices, but also in Java Cards based in [18] which is a very friendly way.

### 3.4.2 U-Prove

The implementation of U-Prove comes from Microsoft and is based on public key cryptography, elliptic curves and hash functions. There are three participants, the user, the issuer and the verifier. There are two protocols:

- **Protocol** 1**:** "The Issuance protocol" which takes place between the user and the issuer and then a cryptographic token is issued and
- **Protocol** 2**:** "The presentation protocol" takes place after the issuance protocol and is between the user and the verifier, where the issued cryptographic token interacts with these participants.

The cryptographic token contains the attributes and cryptographically protects them against tampering, has got a public key and a corresponding private key which are generated during the Issuance Protocol. The token contains the Token Information section which contains metadata and are revealed during the Presentation Protocol and the Prover Information section and is disclosed for the issuer during the Issuance Protocol. The U-Prove provides unlinkability and untraceability because the issuer executed a blind signature over the token and the user demonstrates the possession of undisclosed attributes by executing a zero-knowledge protocol, but the multiple uses of the same U-Prove token are linkable. Also, the revocation system is available because each token has a unique number which is used for tracking revoked tokens. each user may have more than one U-Prove token.

### 3.4.3 I Reveal My Attributes (IRMA)

Moreover, there is the implementation of IRMA from Radboud University of Nijmegen, in [17]. IRMA is based on non-trivial cryptography for attribute-based credentials which are data that contain a provable claim. To achieve this goal, IRMA uses both the technologies of IdeMix and U-Prove, so there are the same protocols and the same participants as there are into the IdeMix and in the U-Prove systems. IRMA is an attribute based identity system and just reveals user's attributes to complete online or offline transactions, these are attributes which are non-identified.

The user's attributes are stored on a personal smart card which is called IRMA card, it contains the picture and the attributes of the user. Also a PIN is required for its use. This IRMA card is issued by an issuer and a user must be authenticated to that issuer and then the issuer will issue this IRMA card for the known user's attributes. After this event, the user can demonstrate this card to a SP, who will check the expected attributes and the user will agree to disclose these attributes to the SP. However, the user must prove that she really owns that IRMA card, so she must enter the PIN code to activate the card for the attributes' disclosure. The security of IRMA is based on the protection of each user's private key, this private key must be safe under the exclusive user's control. These goals can be totally achieved by smart cards which are created in a special way to protect the private key and this will never leak from the card or from other channels.

### 3.4.4 ABC4Trust

Furthermore, there is another implementation which is called ABC4Trust. The goal of ABC4Trust is to provide an implementation of attribute based credential systems where accredited members of communities will provide their members with AC service. To achieve this goal, ABC4Trust uses both IdeMix and U-Prove technologies. ABC4Trust provides strong authentication as well as security based on [19]. Users based on their attributes obtain cryptographic token which contains these attributes and is unlinkable. The token reveals the needed information with the user's permission.

## 3.5 Limitation

Reading all the above information we can understand exactly how the Identity Mixer works. Being more specific, the paper of Jan Camenisch and Anna Lysyanskaya in [13], is very technical and very mathematical. Also, the recent paper under the name Specication of the Identity Mixer Cryptographic Library Version 2.3.0* in [20] it is also very technical but gives some more help with the given pseudo-code. The limitation to these evolutionary papers is that they are not comprehensible for people who are not expert in security, mathematics and the cryptographic sector. We believe that an approach which is more understandable to people who own knowledge of more scientific domains and not only of these three scientific sectors can attract many more people to this interesting scientific domain. So, in our Master Thesis we want to rewrite and explain all these important meanings to the people who want to learn and gain an experience about the AC.

# Chapter 4

# Anonymous Credential

ACs are the way to move into total anonymity and into ABAC systems. In the following sub-chapters, we will explain how an AC can be generated and also we will provide each protocol which is needed to produce and use such an AC.

Based on [10], [21] and [22] the ACs have four (4) basic indispensable properties:

- The first one is that the user should be able to choose what information will be disclosed and can be achieved as it is proposed by David Chaum in [23].
- The second one is that the AC must be hard to forge, so nobody is able to copy an AC or issue an AC without the IO's permission. The security against this threat comes from the digital signature cryptographic technique based on [12].
- The third one is that the transactions must be unlinkable. To achieve this the user must obtain several signatures from the IO, based on [13], [10] and [24].
- The last one is that the AC has to be revocable. To achieve this goal an anonymity revocation manager can be added who will be trusted to locate the user's pseudonym and user's identity. The revocation of AC is decided if there is an important reason because the user's anonymity will be lost after the revocation.

In addition, there are some extra properties which are more theoretical, namely the user should not share her pseudonyms and credentials with other people and the ACs should be one-show credentials.

Nevertheless, there are vulnerabilities in the ACS, based on [22], which are:

- Initially, the most important vulnerability is the revocation because the user is anonymous until the Revocation Authority reveals her identity.
- Secondly, the issuer has to be sure about the attributes that signs at the end of the issuing protocol that these are true and
- Thirdly, the user does not try to cheat on the issuer

## 4.1 Communication Model

To implement an ACS two fundamental protocols are required:

- The **issue protocol** is the first protocol. This protocol is executed between the user and the IO, where the user gives her personal data to the issuer, who checks if these data are true or not and then the issuer returns to the user her AC based on the issuer's public key corresponding to issuer's unique secret key. Each AC is valid under the IO's public key.
- The second protocol is the **show protocol**. The show protocol, on the other hand takes place between the user and the SP, where the first one acts as a prover and the latter as a verifier, i.e. the user proves to the SP with the issued AC that she can use the provided service because she meets the system's requirements. Each SP sets a group of requirements which each user has to satisfy to be able to use the desirable service.

After the AC's issuance, the user will send it to the SP, so she will be checked if she is able to use or not the desirable service. This AC contains the user's information as we said above, and these are cryptographically stored, so the information which is needed from the SP's access policy will not be disclosed as a plaintext. These proofs come from a legit IO who checked the user's information and based on these issued the user's AC. From this protocol, no information is leaked but only the truth of the statements.

For example, if an application allows only users who are over eighteen (18) years old and the user is twenty-one (21) years old, this credential will not reveal the real age but will just ensure that the user is over eighteen years old. So, the user will gain access to the application without revealing his real age and other personal data. This system cannot only be used into computers, but also in smart devices i.e. smartphones and smartcards.

The communication model of an ACS among the three participants (i.e., IO, user and SP) is depicted in Figure 4.1. Initially, the IO owns only his key pair which consists of a public key (pK) and a secret key (sK), the user owns only her attributes and the SP owns only his pK.

The **issue protocol** is composed of a two-round interaction:

**Step 1:** At the start, the user sends her attributes to the issuer and

**Step 2:** secondly the IO checks the attribute's validity to generate and send the user's AC to the user.

Also, the **Show Protocol** consists of two steps:

**Step 1:** where at first the user sends her AC to the SP and

**Step 2:** secondly the SP checks the statements to let her use the service or reject her.

The issued ACs are valid only under the issuer's public key and only the issuer knows its counterpart secret key [21], [1], [2]. This happens because the verifier who will receive the credential will check that the credential comes from a legal IO who "signed" it with his specific secret key. The IO signs the credentials with his secret key so each verifier can be assured that the credential is issued from an IO and the user didn't cheat, as we said above the credential must not be forged. This can be checked with the use of IO's public key.

Furthermore, the AC does not transmit the credential itself but transmits only proofs to convince the verifier that the user is able to use the service. This action happens without leaking anything about the credential other than the shown properties. This fact leads to the advantage that the SP cannot reuse the AC and the user's information to pretend the user. ACs also have one more advantage, that the user can choose the subset of the disclosed attributes. Also, the Show Protocol allows to prove statements from multiple ACs at once from the same user.

From the above description, we understand that trusted parties are needed from both user, IO and SP part. This happens because all the participants want to ensure that the system will work fine and no evil or liars will be hidden into this system. Also, they want to be sure that the other part follows the instructions, so the trust between the participants is important. We should mention the if the SP does not follow the ACS's rules then the user can ignore him and choose another provider, but if the SP wants to be a legit part of this system then she must follow the rules and not try to cheat. Furthermore, trust between the IO and the SP is required. However, this kind of trust cannot be gained with an online protocol or a method but only with the reputation which an IO can gain through the quality of service which he provides. There is the situation where the IO and the SP is the same organization, then the same terms apply. They must not try to blend their knowledge which they have acquired from the common customers who the own to find out customer's information because this leads to the system's infringement and then the user will lose her trust to the system and the anonymity will collapse. The following figure, Figure 4.1 demonstrates fully the communication model and also contains the trust ties which bind the participants, whether the tie is achieved with online or with offline transaction.

## 4.2 Anonymous Credential Protocols

The two basic protocols which are used are:

- **The issuing protocol** which takes place between the user and the IO where the user becomes a recipient and waits for the IO to issue a

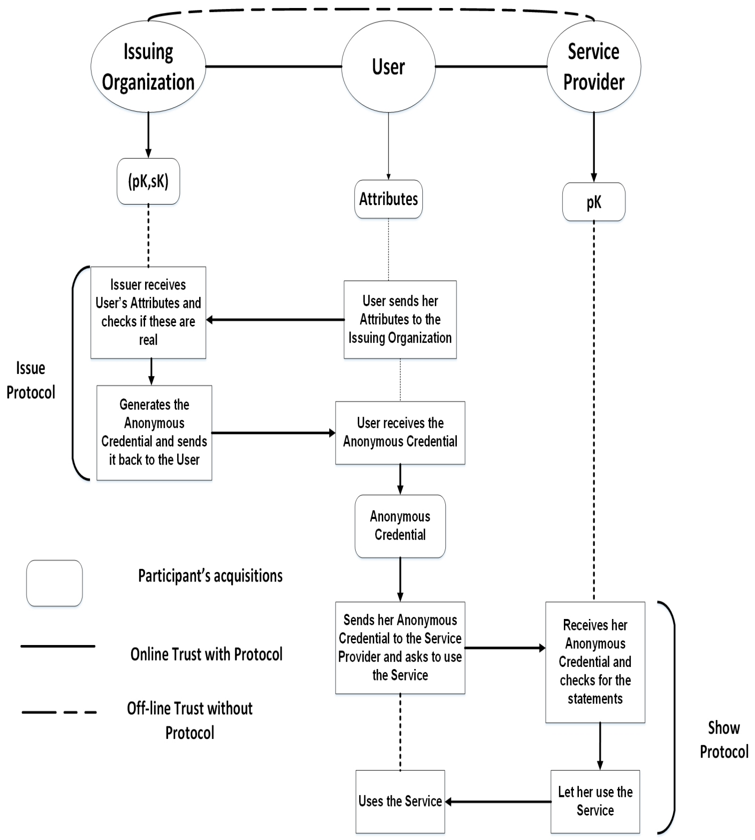

Issuing Organization
User
Service Provider
(pK,sK)
Attributes
pK
Issuer receives User's Attributes and checks if these are real
User sends her Attributes to the Issuing Organization
Issue Protocol
Generates the Anonymous Credential and sends it back to the User
User receives the Anonymous Credential
Anonymous Credential
Participant's acquisitions
Online Trust with Protocol
Off-line Trust without Protocol
Sends her Anonymous Credential to the Service Provider and asks to use the Service
Receives her Anonymous Credential and checks for the statements
Show Protocol
Uses the Service
Let her use the Service

Figure 4.1: Anonymous Credential Communication Scheme

credential for her. The first protocol (i.e., issuing protocol) has two (2) sub-protocols which are:

  – **The generation of a pseudonym**, which takes place between the user and the IO and its scope is to create a pseudonym for the user and

  – **The generation of a credential**, which takes place between the user and the IO and intends to create a credential based on a specific pseudonym, i.e. based on the pseudonym created in the previous step.

- **The show protocol** which takes place between the user and the SP, where the user becomes a prover to prove that she owns the right to use the service as it is mentioned above. The second protocol (i.e., show protocol) also consists of two (2) other sub-protocols which are called:

- **The showing a single credential**, which takes place between the user and the SP where the user wants to prove to the verifier that she owns a unique credential which is generated from a specific IO and
- **The showing a credential respect to a pseudonym**, which takes place between the user and the SP where the user wants to prove to the verifier that she owns a unique credential which is generated from an specific IO based on a unique pseudonym and these are based on the same master secret key.

In the following table, Table 4.1, are defined the notations that are used for the proposals analysis.

| Notation | Definition |
|---|---|
| $O$=$O_i$ | IO |
| $O_j$ | Verifier/Service Provider |
| $PK_O$ | Organization's public key |
| $n_O$ | Strong RSA modulus |
| $QR_{n_O} : (a_O, b_O, d_O, g_O, h_O)$ | Quadratic Residue modulo $n_O(QR_{n_O})$. This means that a number $q$ is $(QR_{n_O})$ if there is a number x such that $x^2$=$q(modn_O)$, otherwise the q is called quadratic non-residue modulo $n_O$. The values $a_O, b_O, d_O, g_O, h_O$ are quadratic residue. |
| $U$ | User |
| $X_U$ | User's masters key |
| $N_{(U,O)}$ | Pseudonym between a user and an organization |
| $P_{(U,O)}$=$a_O^X \cdot b_O^{S_{(U,O)}}$ | Pseudonym's validating tag |
| $S_{(U,O)}$ | Random string whose parts are generated both from the user and from the IO. The final value is known only to the user. |
| Credential | Tuple of ($e_{(U,O)}$,$c_{(U,O)}$) |
| $e_{(U,O)}$ | Long prime |
| $c_{(U,O)}^{e_{(U,O)}} \equiv P_{(U,O)} \cdot d_O$ | |
| $l_n$ | The length of all RSA moduli |
| $\Gamma$=$-2^{l_\Gamma}, 2^{l_\Gamma}$ | Integer Interval |
| $\Delta$=$-2^{l_\Delta}, 2^{l_\Delta}, l_\Delta$=$e(l_\Lambda+l_n)+1, e > 1$ | Integer Interval |
| $\Lambda$=$2^{l_\Lambda}, 2^{l_\Lambda+l_\Sigma}, l_\Lambda > l_\Sigma + l_\Delta + 4$ | Integer Interval |

Table 4.1: Notation and Definition Table

### 4.2.1 Protocol 1.1: The Generation of a Pseudonym

This protocol, as it is proposed by [13], is the first interaction between a user and an IO generating a pseudonym that will be used in the future communications of the user with SPs. Pseudonyms are unique, and one pseudonym will be used by only one user. Moreover, a user may have a lot of pseudonyms (user's public keys), where each one can be used more than one times. This protocol takes place between the user and the IO, where the user establishes a pseudonym $N_{(U,O)}$and its validating tag $P_{(U,O)}$, where the IO checks if the pseudonym's validating tag is in the right form. This happens because IO needs to check if the validating tag is alright. This is important because this value shall play a significant role in the future steps for the pseudonym's and protocol's creation. A wrong form of this value has notable impact because the protocol must restart which is a time-consuming procedure, also each IO may have a different form for the validating tag, so a validating tag from one another IO cannot be used twice or more times. One validating tag is valid for only one session. This operation is completed in seven (7) steps as depicted in Figure 4.2.

**Step 1:** This step takes place on the user's side. We have to mention that the pseudonym consists of two parts which will be randomly chosen by the user and the IO respectively. Initially, the user chooses randomly the first part of the pseudonym, $n_1$, which is element of the $\{0,1\}^k$ which is an integer with the length of k. Also, she randomly chooses the value of $r_1$ which is an element of the $\Delta$ integer interval and the values of $r_2, r_3$ which are real integers of $\{0,1\}^2 l_n$ with length of $2l_n$. Subsequently, the user computes the values of $C_1$=$g_O^{r_1} \cdot h_O^{r_2}$ and $C_2$=$g_O^{X_U} \cdot h_O^{r_3}$ using the random values of $r_1, r_2, r_3$ and some parts of IO's public key. At last, user sends to the IO the values of $N_1$,$C_1$ and $C_2$.

**Step 2:** This step happens in both sides because the user must prove to the IO that the values of $C_1$ and $C_2$ are correctly formed, to be accepted. To prove that $C_1$ and $C_2$ are correctly formed, the user uses the theory of ZKP and creates some secret values, which are known only to herself. User's goal is to prove that the values of $C_1$ and $C_2$ are equal to modulus which contain both the secret values and part of IO's public key. Only if this proof is real, will the protocol continue from the issuer Organization's side.

**Step 3:** This step will occur only if the above proof was successfully completed and takes place on IO's side. The IO chooses the random value of r which is an element of the  integer interval. Moreover, the IO chooses the value of $N_2$ which is an element of the $\{0,1\}^k$, is an integer with the length of k and is the second part of the pseudonym. Then, the IO sends these two values to the user.

**Step 4:** The fourth step is executed on the user's side. User knows $N_1$

and $N_2$, so she is ready to create the pseudonym. To create the pseudonym, $N_{(U,O)}$, user just concatenates these two values, so $N_{(U,O)} := N_1||N_2$. Furthermore, the user based on the value of $r_1$ from the rst step and the received value of r from the step 3 and creates the secret value of $s_{(U,O)} = (r_1 + rmod(2^{l_\Delta+1} - 1)) - 2^{l_\Delta} + 1$. Also, the user computes the value of the validating tag, $P_{(U,O)} := a_O^{X_U} \cdot b_O^{s_{(U,O)}}$, based on the user's private key,$X_U$, the secret value of $s_{(U,O)}$and some parts of IO's public key,$a_O$ and $b_O$. After this computation the user sends the validating tag,$P_{(U,O)}$to the IO.

**Step 5:** This step happens in both sides because the user must prove to the IO that the pseudonym's validating tag is correctly formed. So, at first, the user randomly chooses the value of $r_4$ which is an element of the $\{0,1\}^{2l_n}$and computes the value of $C_3$ based on the secret value of $s_{(U,O)}$, the $r_4$ and some parts of IO's public key. After this computation, the user sends the value of $C_3$ to the IO. Also, the user proves to the IO that the validating tag is correctly formed. To prove it, the user uses the theory of ZKP and creates some secret values which are known only to herself. The user's goal is to prove that the value of $P_{(U,O)}$ is equal to modulus which contains both the secret values and some of IO's public key, and that the values of $C_1$,$C_2$ and $C_3$ are equal to modulus which contains both the secret values and some of IO's public key. Only if this proof is real, does the protocol continue to the IO's side and user's side otherwise the protocol will be canceled.

**Step 6:** The IO stores the values of pseudonym, $N_{(U,O)}$, of validating tag, $P_{(U,O)}$ and of squared validating tag, $P^2_{(U,O)}$.

**Step 7:** The user stores the values of pseudonym, $N_{(U,O)}$, of validating tag, $P_{(U,O)}$, of squared validating tag, $P^2_{(U,O)}$ and the secret value of $s_{(U,O)}$.

### 4.2.2 Protocol 1.2: The Generation of a Credential

The second Protocol called "Generation of a Credential", takes place between the user and the IO and its basic goal is to issue a credential for the user. Being more specific, the IO uses the values of the pseudonym and the validating tag which are created in the previous Protocol 1.1 and are sent by the user to create the credential. The pseudonym binds on the AC. The credential's generation is the most important part of ACS because the user

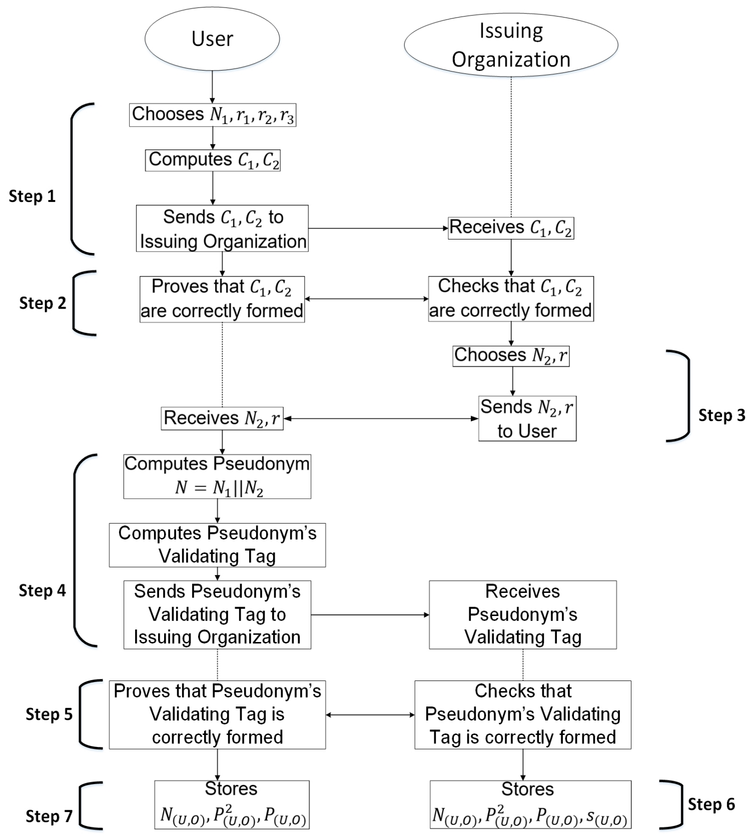

User
Issuing Organization
Step 1
Chooses $N_1, r_1, r_2, r_3$
Computes $C_1, C_2$
Sends $C_1, C_2$ to Issuing Organization
Receives $C_1, C_2$
Step 2
Proves that $C_1, C_2$ are correctly formed
Checks that $C_1, C_2$ are correctly formed
Chooses $N_2, r$
Step 3
Receives $N_2, r$
Sends $N_2, r$ to User
Step 4
Computes Pseudonym $N = N_1||N_2$
Computes Pseudonym's Validating Tag
Sends Pseudonym's Validating Tag to Issuing Organization
Receives Pseudonym's Validating Tag
Step 5
Proves that Pseudonym's Validating Tag is correctly formed
Checks that Pseudonym's Validating Tag is correctly formed
Step 7
Stores $N_{(U,O)}, P^2_{(U,O)}, P_{(U,O)}$
Stores $N_{(U,O)}, P^2_{(U,O)}, P_{(U,O)}, s_{(U,O)}$
Step 6

Figure 4.2: Actions of protocol 1.1

will give it to the SPs to use their service. This credential contains information which come from the user but also come from the IO. This protocol is completed in three (3) steps which will be analyzed below and depicted in Figure 4.3.

**Step 1:** During the first step, which takes place on the user's side, the user sends to the IO his pseudonym and the validating tag from the previous protocol. Also, the user proves that these elements belong to herself. To achieve it, the user whereby the theory of ZKP creates some secret values which are known only to herself. The user's goal is to prove that the squared value of $P_{(U,O)}$ is equal to modulus which contains both the secret values and some elements of IO's public key. Only if this proof is real, does the protocol continue on the IO's side.

**Step 2:** In the second step, which is executed on the IO's side, the IO creates the user's credential. So, at first, the IO checks that the received validating tag and pseudonym are stored in his database. If this check is true, then the protocol will continue its execution. The IO randomly chooses a big prime,$e_{(U,O)}$,and computes the credential, $c_{(U,O)}$ based on some elements of his public key, the validating tag and the prime $e_{(U,O)}$. Also, the IO stores the outcome of this computation in its database for this specific pseudonym. At last, the IO sends the credential, $c_{(U,O)}$, to the user.

**Step 3:** In this step which happens on the user's side, the user stores the received credential, $c_{(U,O)}$, into her database. Before it starts, the user will check if the power of $c_{(U,O)}$ and $e_{(U,O)}$ is equal to $P_{(U,O)} \cdot d_O(mod, n_O)$, if it's true then the user stores the tuple of $(c_{(U,O)}, e_{(U,O)})$ for this specific IO. Furthermore, the credential record consists of the pseudonym $P_{(U,O)}$, the credential $c_{(U,O)}$ and the big prime $e_{(U,O)}$.

### 4.2.3 Protocol 2.1: The Showing a Credential

The third protocol is called, "Showing a Single Credential" and takes place between the user and the SP where the user needs to prove to the SP that she owns a credential which is issued by an IO. Being more specific the user has to prove based on the protocol 1.1 and protocol 1.2 again to the SP. This protocol is completed in just two (2) steps, which we will analyze below.

**Step 1:** At this step, the user chooses two random numbers $r_1$ and $r_2$ and computes the value of $A = c_{(U,O)}{}^{r_1}_{h_O}$ and the value of $B = h_O^{r_1} \cdot g_O^{r_2}$. After the computations the user sends these values, A and B, to the verifier.

**Step 2:** This step takes place in both sides and the user proves to the verifier that she owns a credential which is issued by the IO and is registered on some pseudonym with the same IO. To achieve this goal, the user uses the theory of ZKP and creates some secret values which are known only to herself. In these comparisons only, the secret values of ZKP take place, the values which are sent and public key elements of IO. If these comparisons are successfully completed, then the verifier will accept the user.

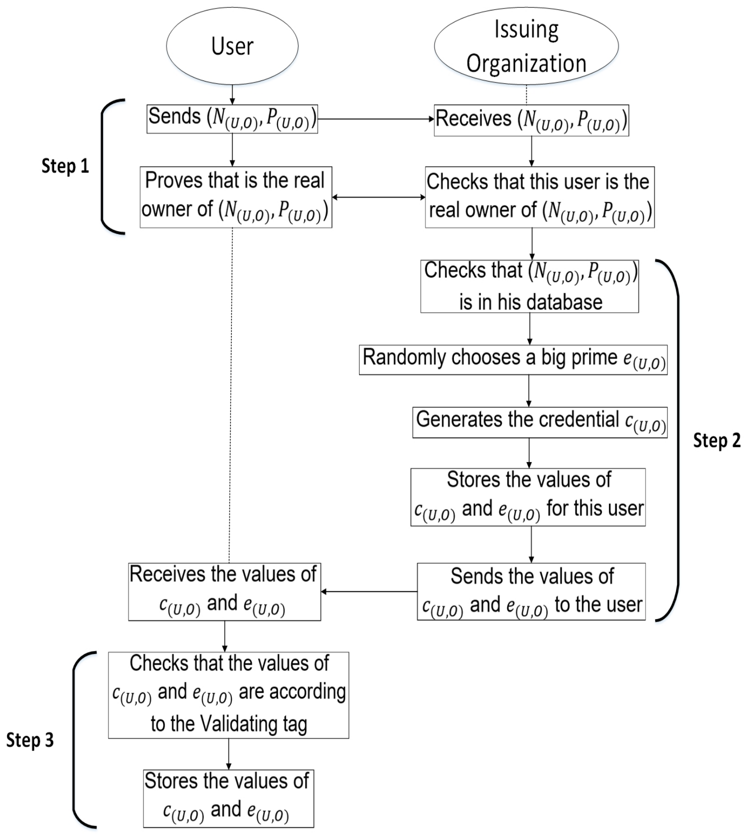

User
Issuing Organization
Sends $(N_{(U,O)}, P_{(U,O)})$
Receives $(N_{(U,O)}, P_{(U,O)})$
Step 1
Proves that is the real owner of $(N_{(U,O)}, P_{(U,O)})$
Checks that this user is the real owner of $(N_{(U,O)}, P_{(U,O)})$
Checks that $(N_{(U,O)}, P_{(U,O)})$ is in his database
Randomly chooses a big prime $e_{(U,O)}$
Generates the credential $c_{(U,O)}$
Step 2
Stores the values of $c_{(U,O)}$ and $e_{(U,O)}$ for this user
Receives the values of $c_{(U,O)}$ and $e_{(U,O)}$
Sends the values of $c_{(U,O)}$ and $e_{(U,O)}$ to the user
Checks that the values of $c_{(U,O)}$ and $e_{(U,O)}$ are according to the Validating tag
Step 3
Stores the values of $c_{(U,O)}$ and $e_{(U,O)}$

Figure 4.3: Actions of Protocol 1.2

We can also see the successful flow of Protocol 2.1 in steps from the Figure 4.4 which demonstrates the actions which are taken by both participants.

### 4.2.4 Protocol 2.2: The Showing a Credential with Respect to a Pseudonym

The fourth Protocol is called, "Showing a Credential with respect to a pseudonym" and takes place between the user and the SP. That protocol looks like the third protocol. Being clearer, the user at this time must prove to the SP,$O_i$, that she owns a credential established by an IO,$O_j$, with whom the user has established a pseudonym, but also the user must prove that the

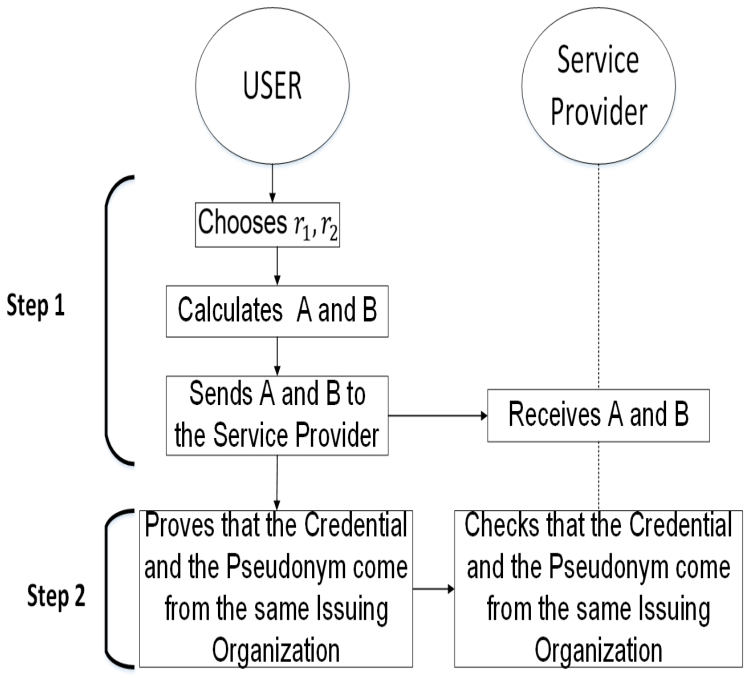

USER
Service Provider
Step 1
Chooses $r_1,r_2$
Calculates A and B
Sends A and B to the Service Provider
Receives A and B
Step 2
Proves that the Credential and the Pseudonym come from the same Issuing Organization
Checks that the Credential and the Pseudonym come from the same Issuing Organization

Figure 4.4: Actions of Protocol 2.1

pseudonym and the credential come from the same secret key. This protocol is completed in just two (2) steps, which we will analyze below.

**Step 1**: This step takes place on the user's side, where the user chooses the random numbers $r_1$,$r_2$,$r_3$ and computes the value of $A = c_{(U,O_j)}{}^{r_1}_{h_{O_j}}$ and the value of $B = h^{r_1}_{O_j} \cdot g^{r_2}_{O_j}$. After the above computations the user sends these values, A, B, and the pseudonym,$N_{(U,O_j)}$,to the verifier, $O_i$.

**Step 2:** This step is executed in both sides, because the user must prove to the SP that the credential and the pseudonym come from the same master secret key. To achieve this, the user whereby the theory of ZKP creates some secret values which are known only to herself. During the comparisons only, the secret values of ZKP take place, values which are sent and public key elements of IO and of SP. The computations/comparisons which take place in this step are similar to the second step of the Protocol 2.1, the only difference is a new comparison which will prove the use of the same master secret key.

We can also see the successful flow of Protocol 2.2 in steps from the Figure 4.5 which demonstrate the actions which are taken by both participants.

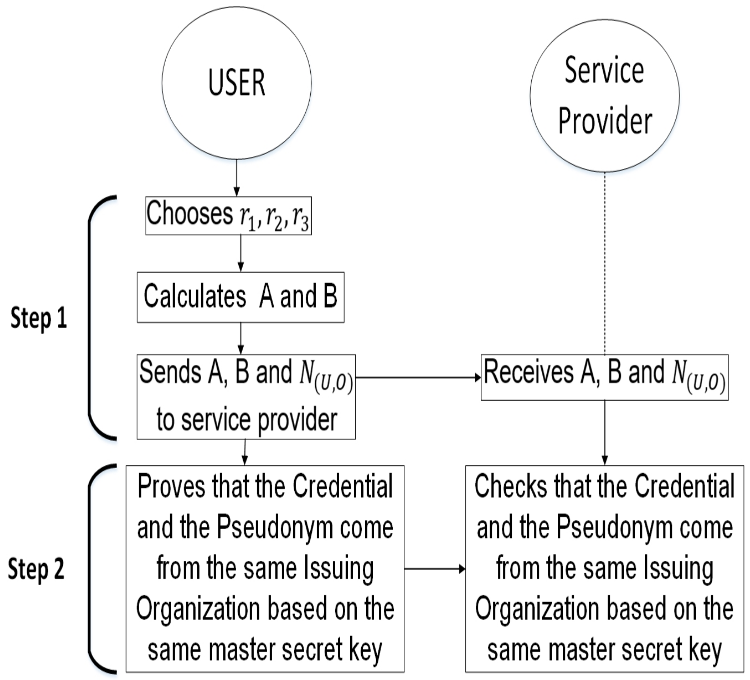

USER
Service Provider
Step 1
Chooses $r_1, r_2, r_3$
Calculates A and B
Sends A, B and $N_{(U,O)}$ to service provider
Receives A, B and $N_{(U,O)}$
Step 2
Proves that the Credential and the Pseudonym come from the same Issuing Organization based on the same master secret key
Checks that the Credential and the Pseudonym come from the same Issuing Organization based on the same master secret key

Figure 4.5: Actions of Protocol 2.2

# Chapter 5

# ReCRED

The purpose of the ReCRED project (From Real-world Identities to Privacy-preserving and Attribute-based CREDentials for Device-centric Access Control),[25], is to forward the user's personal mobile device to the role of a unified authentication and authorization proxy towards the digital world. Also, ReCRED faces daily common security and private issues such as the loss and theft of a mobile device. To achieve these goals, ReCRED uses user-to-device authentication and device-to-service authentication mechanisms, multi-factor authentication mechanisms based on behavioral, physiological user signatures and privacy awareness tools.

## 5.1 Architectural Overview

There are five (5) participants in ReCRED, who are the User Device (UD), the Identity Consolidator (IDC), the Service Providers (SP), the Behavioral Authentication Authorities (BAA) and the Identity Provider (IDP). We will explain their role below.

**UD** is the central and the most important component of the ReCRED project's architecture given the fact that the ReCRED intends to provide device-centric authentication. For an authentication between the user and the SP, the user at first must authenticate to her device using bio-metrics. Moreover, ReCRED deploys anonymous cryptographic credential methods, which are the IdeMix and the U-Prove, on the device to enable Privacy-preserving Attribute Based Access Control (PABAC). Using these methods the users are able to request from IDCs and IDPs the issuance of cryptographic credential based on these methods. Furthermore, these methods will disclose the ACs to the IDPs during the authentication. The credentials which come from these methods will be stored into the Cryptographic Credential Storage (CCS) which is a UD's part. The credentials which are stored into the CCS must not be exported even if the UD is compromised. In addition, to enable the second factor authentication the UD contains a module which monitors the user's behavior from many sensors available in the device.

**IDC** is another major participant of the ReCRED architecture and participates in the most use cases of the ReCRED platform. The IDC collects identity attributes from the IDPs upon user's request and these are securely stored in an ID repository. Also, IDC provides fail-over authentication mechanism in cases where the user loses access to her device. IDC encapsulates the FIDO (Fast IDentity Online) protocol, which allows the IDC to act as an OpenID Connect (OIDC) provider. This protocol allows to exchange verified identity attributes with the IDP. The users of ReCRED through the IDC can manage their IDC account. IDC is responsible to "control" the users' account and lock them if there is a compromise on these accounts. IDC runs IdeMix and U-Prove therefore it enables the ReCRED to contain PABAC in the architecture. The issued credentials can be backed-up in the IDC for failure recovery purposes. In addition, the IDC allows users to exchange their attributes between different IDPs through the OIDC.

**IDP** contains a ReCRED daemon that provides him with the ability to store user's identity attributes in identity repositories. Furthermore, IDPs contain cryptographic credential methods, i.e. IdeMix, to issue ACs to the users based on the user's identity attributed which are stored into its idenity repository. The credentials issued by the IDPs are also transferred to the IDC to store them so that they are backed up and accessible if the IDC fails. Furthermore, the UP or the IDC communicates with the IDPs using federated log-in protocols, i.e. OIDC.

**SP** is the verifier who provides users with some services. To achieve this, they have installed a ReCRED daemon to their system. The RecRED uses the OIDC protocol, so in this case the SP is the Relying Party in order to communicate with other entities in the architecture. Moreover, the SPs are able to support FIDO and PABAC without the IDP's interference during the authentication process. After the FIDO authentication,[27], the UD transfers the FIDO credentials to the verifier using public key cryptography. Also, the verifier can ask for a second factor authentication from the user. If this happens, then the SP redirects the user's device to the IDC and the IDC will notify the SP which authentication authority should contact for each second factor authentication.

**BAA** are autonomous entities and are responsible for capturing and maintaining the behavior of the users on their devices. Also, they are responsible for performing both on-demand and continuous behavioral second-factor authentication. The BAAs to achieve this the "monitor" the user's behavior. Furthermore, they can store on their behavioral profile database, the behavior of the user. Using the behavioral information, they can determine whether a device currently behaves as it usually does. Depending on the result, they can verify to the verifier whether he believes the device is held by its legitimate user. The results of the BAA's opinion is released to the IDCs or SPs using the OIDC protocol.

The ReCRED supports and provides ABAC system that lets the system handle ACs and it is based on the technologies of IdeMix and U-Prove. However, these are the core of PABAC system, but have different requirements for the interfaces. To provide a common interface between the technologies of IdeMix and U-Prove the ReCRED used the FIWARE project, which will let the ReCRED provide support for both IdeMix and U-Prove credentials support and pair together the results of FIWARE and IRMA projects. Furthermore, the ReCRED project will use the FIWARE project for the verification and issuance of credentials between the user device and the API Server.

## 5.2 ReCRED-Anonymous Credential Implementation

One of the most important goals of the entire ReCRED project is to provide full protection of the privacy of the individuals who use the services.

In order to achieve this key goal, the ReCRED uses a private credential approach to let users prove their identity attributes safely from their device to the online or physical relying service. Actually, in most real-world scenarios, users do not need to disclose their full identity as the SPs require knowing only an aspect of their identity (i.e. their age, their home address, their profession, whether they are students).

However, users are compelled to disclose their full identity to the SPs. By using ABAC, our solution becomes privacy-preserving by design thanks to the integration of ACS, like IdeMix and U-Prove. Also, the ReCRED also uses the technology of IRMA, ABC4Trust and FIWARE. The final ReCRED's target is to integrate all the above technologies to provide an integrated system.

### 5.2.1 ReCRED and IdeMix

IdeMix,[26], is a very important function of ReCRED project because it will enable the ReCRED framework to support anonymous, unlinkable and untraceable credential issuing and proving without disclosing any personal data, [20]. IdeMix code can be downloaded for free from [28]. Also, IdeMix prototype is provided which is the basic implementation of the IdeMix instances which will be used at the UD, authorities, IDC and online services components. Moreover, ReCRED project has already created services with an API, which allows the IdeMix transactions. In this implementation the user is operated as an HTTP client, while the issuer and the verifier as java-based web services.Furthermore, ReCRED provides an alternative implementation of IdeMix which is developed in Python, which is called Pydemix. This implementation is available on GitHub [29]. Also, this implementation differs from the other JVM implementations, because this can run in both sides of the user and SP. The Pydemix's cryptographic functions following strictly the IdeMix specification, allowing easy readability and investigation of the correctness of the source code. Likewise, ReCRED provides an IdeMix API Server which communicates with the IRMA API Server to produce the desired token to be provided to the user device and at last the user device receives the IdeMix credential from the IRMA API Server.

### 5.2.2 ReCRED and U-Prove

U-Prove, [30], is another innovative cryptographic technology which is used by ReCRED. U-Prove allows users to minimally disclose certified information about themselves when interacting with online SPs. Moreover, U-Prove provides features of Public Key Infrastructure (PKI) and strong privacy protections by offering user control and preventing unwanted user tracking. ReCRED also provides a U-Prove API Server. The U-Prove API Server 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 output of the issuance process is a U-Prove token. Regarding the U-Prove implementation, it is an open-sourced C# and JavaScript implementation, while having a legacy Java version. To preserve the compatibility with the other ReCRED technologies the U-Prove implementation will be exposed as a web-service. 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's side, when issuing a token. The device acts as a U-Prove token extension, having a private key.

### 5.2.3 ReCRED and IRMA

The IRMA project,[31], which is used by the ReCRED aims at providing an open sourced, secure, decentralized and easy way to use implementation of PABAC with minimal disclosure of attributes for online and offline transactions. Also, IRMA project [32] provides an open source IdeMix intending to use the implementation targeted at mobile devices. Moreover, IRMA provides its use through smartcards. ReCRED intends to use the IRMA card emulator as a cryptographic credential storage as an alternative to the Trusted Execution Environment. Furthermore, the ReCRED provides an IRMA API Server. This is a server which sits between the user device and the service or the IDPs on the other side. Also, it is responsible for the IdeMix credential issuance. It handles all specific cryptographic details of issuing credentials and verifying disclosure proofs on behalf of the service or IDP.

### 5.2.4 ReCRED and FIWARE

Furthermore, FIWARE, [33], is used by ReCRED project. FIWARE is an open platform that aims to provide a novel service infrastructure to an easier development of Internet applications. The ReCRED's goal through the FIWARE interface is to enable the seamless use of the IdeMix and U-Prove. FIWARE security specifications are based on the ABC4Trust specifications [34] which propose a cryptographic agnostic attribute credential protocol, thus supporting both IdeMix and U-Prove.

### 5.2.5 ReCRED and ABC4Trust

At last, the ABC4Trust project is used by ReCRED. ABC4Trust offers an implementation of a privacy-preserving attributed based credential engine which is protocol agnostic: the p2abc project. The FIWARE project built an additional layer on top of the p2abc project offering a privacy FIWARE GE (Generic Enabler): the p2abc-zhaw project, which is available in [35]. The latter project uses the Microsoft C# based on U-Prove implementation. Moreover, there is the ABC4Trust prototype which the p2abcengine source code, which is available for free in [36], and provides a set of core components useful to set up an attribute based authentication ecosystem. The available components can be deployed in two ways: running independent web services that already expose RESTful interfaces, or integrating the code directly into a Java code-base. Both ABC4Trust and FIWARE can be used as web services and the implementation is Java based (the deployment environment could be Tomcat or another servlet engine).

## 5.3 ReCRED - Anonymous Credential: Implementation Guide

In this section we will discuss the currently available implementations about the AC which are provided by the ReCRED project. ReCRED project produced AC implementations which are about the device application which is called ReCRED device wallet and the U-Prove implementation in android environment. In the following paragraph we will analyze the importance of these implementations, how these can be set up and how these can be used in real life. Also, in the next chapter we provide two use cases which will explain how the whole ReCRED can be used in everyday life.

### 5.3.1 ReCRED - Device Wallet

At first, we will analyze the ReCRED Device Wallet. The ReCRED Device Wallet is designed to manage the user's credentials and support the user during the issuance and verification. Furthermore, all the issued credentials will be available through this application. The first screen when we open the application is demonstrated in the Figure 5.1. We can see that there is

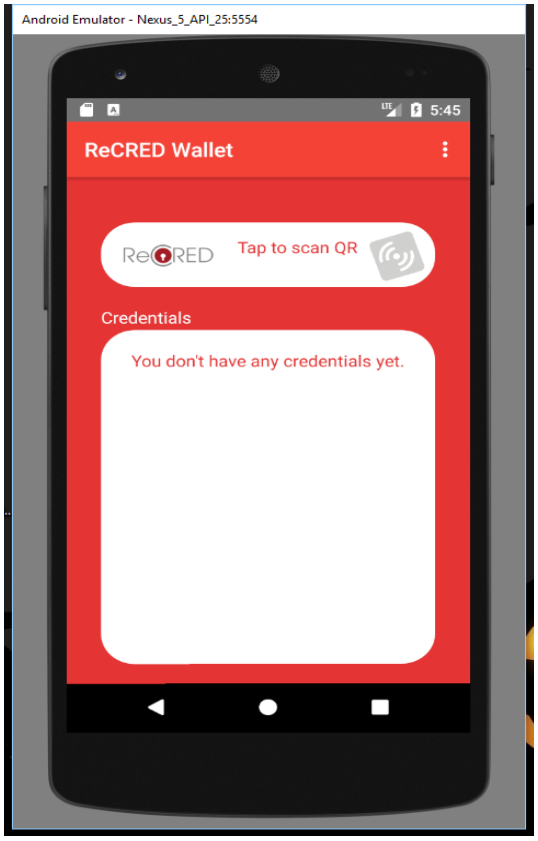

Android Emulator - Nexus_5_API_25:5554
5:45
ReCRED Wallet
ReCRED
Tap to scan QR
Credentials
You don't have any credentials yet.

Figure 5.1: First-Screen of ReCRED Device Wallet

an option for the device to scan a QR-code. This QR-code will provide the credential into the application. When the user presses the button "Tap to scan QR" then the device will search for QR-code and then it will execute the issuance protocol. Subsequently, the user will be asked to provide her consent to provide her credentials and to receive the credential based on these attributes, the Figure 5.3 demonstrates this event. The user is able to accept or decline this question, if the user accepts the offer then the credential will be stored into the device, if not, then the procedure will be stopped. When the credential's issuance is completed then the application looks like the Figure 5.4. The Figure 5.2 demonstrates how the application tries to scan a QR-code.

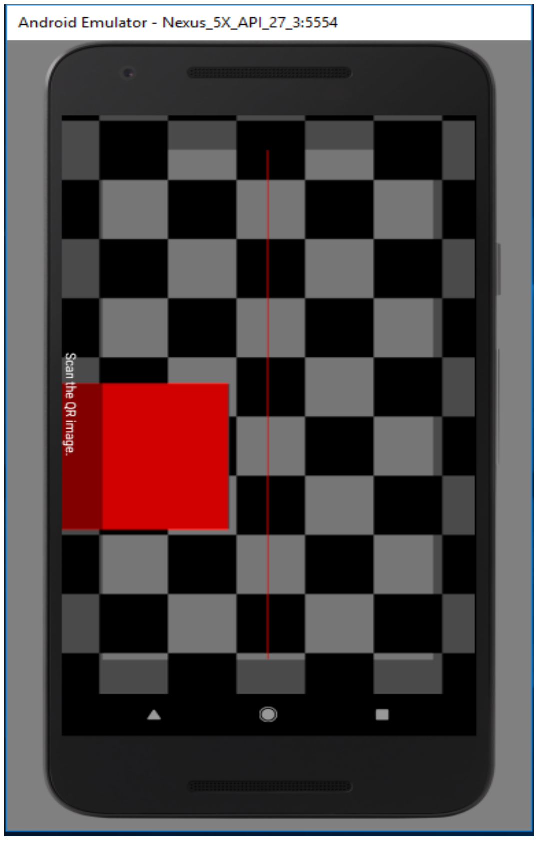
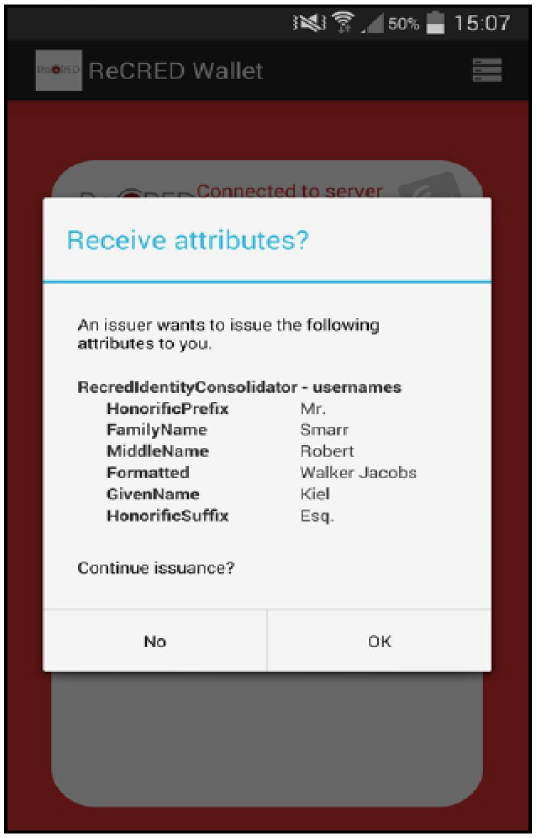


Figure 5.2: Scan a QR-code

To set up the ReCRED Device Wallet we used a linux environment which is Ubuntu 16.04 LTS which is available for free in [37]. The Android studio which is available for free in ,[38]. It's necessary to install the late version of java, gradle and maven. To install java you should follow the next steps:



Receive attributes?

An issuer wants to issue the following attributes to you.

RecredIdentityConsolidator - usernames

| | |
|---|---|
| HonorificPrefix | Mr. |
| FamilyName | Smarr |
| MiddleName | Robert |
| Formatted | Walker Jacobs |
| GivenName | Kiel |
| HonorificSuffix | Esq. |

Continue issuance?

Figure 5.3: Issue or not the credential

**Step 1:** sudo apt-get install default-jre

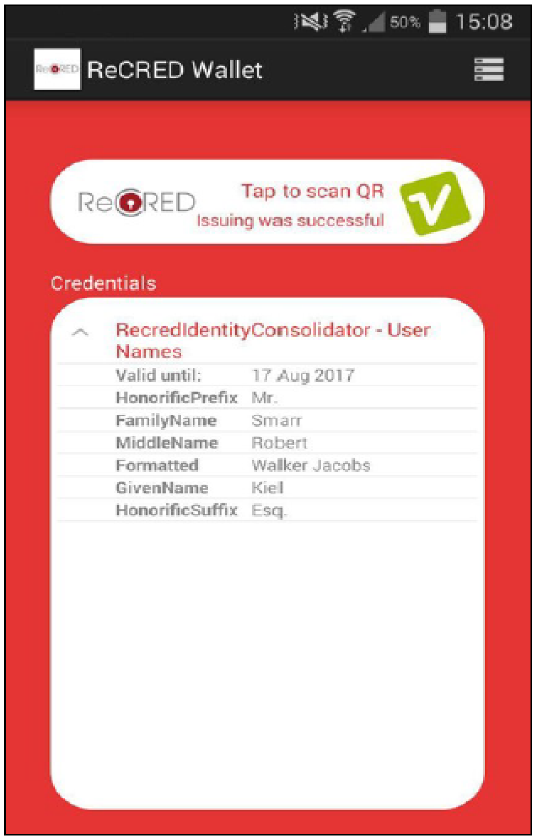
**Step 2:** sudo apt-get install default-jdk

**Step 3:** sudo apt-get install python-software-properties

**Step 4:** sudo add-apt-repository ppa:webupd8team/java

**Step 5:** sudo apt-get update

**Step 6:** sudo apt-get install oracle-java8-installer



15:08

ReCRED Wallet

RecredIdentityConsolidator - User Names

| | |
|---|---|
| Valid until: | 17 Aug 2017 |
| HonorificPrefix | Mr. |
| FamilyName | Smarr |
| MiddleName | Robert |
| Formatted | Walker Jacobs |
| GivenName | Kiel |
| HonorificSuffix | Esq. |

Figure 5.4: Device Wallet from the Credential

To install the Android Studio you should follow extracted the zipped file with name **android-studio-ide-171.4443003-linux.zip** , the name varies based on the IDE's version, into your desirable directory, then execute the command **cd android-studio-ide-171.4443003-linux/android-studio/ bin** and then execute the command **./studio.sh** , after this command you must follow the provided instructions from the interface which will emerge. After these installations the user is able to set up, experiment the provided applications. The android studio is ideal for these actions because it provides solutions to fix the“problems” automatically.

Firstly, we will analyze how a developer can set up the ReCRED Device Wallet application. The folder, under the name ABAC-Idemix contains a folder which is called **recred_ device_ wallet** , this is the folder which we will deal with. The most important phase is the apk's creation, because this phase will produce the apk which will be imported to the user's device. To produce the apk the following command has to be executed, **./gradlew** assemble, in the current folder, **recred_ device_ wallet**. After this apk is produced then you should import it to an android device. The android device could be a physical device or a virtual device. We will use a virtual device, in this device we will import all the apks which we will produce and there is no conflict between them. To execute the apk you should use the android studio which provides the functionality, from the option in the first screen which called Profile or debug APK and then choose the apk with the name **recred_ device_ wallet-debug.apk** . Furthermore, you should press the Run button and then the application will be executed into the device. After this, we will see the first screen which is demonstrated in the Figure 5.1. If we touch the button"**Tap to scan QR**" then we will see the Figure 5.2. This happens because we use a virtual device and because these is no QR-code to scan. A QR-code which could provide us with a credential could lead us to the Figure 5.3 and if the user accepts the question for the credential's issuance this would lead to the Figure 5.4.

### 5.3.2 U-ProveAAR

At this section we will analyze **U-PropveAAR**. The **U-ProveAAR** is the U-Prove implementation for the android device only. This component is more complex than the previous one. At this phase, we need to install maven in our environment, to do this we shall execute the command **sudo apt-get install maven**. After this we should move to the folder **ABAC-Uprove/eu.recred.uprove**. In this folder we will see that there is a **pom** file, this file is important because this will produce our war and files file which are important for the application's execution. Moreover, we should execute the command **mvn install** which will create the jar and war file in the folders which are contained into the **eu.recred.uprove** folder. However, at this time we will use only the jar file which is located into a sub-folder into the **uprove_ engine** folder. To find it we should execute the command **cd eu.recred.uprove/uprove_ engine/target** and then we will find a file with the name **uprove_ engine-0.0.1-SNAPSHOT.jar** and this file we should copy it to the **ABAC-Uprove/UProveAAR/app/libs** , if you don't have a folder with name **libs** into the app folder then you should create one with that specific name, libs.

Moreover, after these actions we should install and set up two (2) **Tomcat** servers which will listen to **two different ports**, one will listen to the 8000 and the other one to the **8001**. To install the tomcat, you should download it from [39]. After this, you should move to the folder **ReCRED-daemons/fiware-api** where there is a pom file, also you should execute the command **mvn install** and then one war file with name **fiwareprivacyrestapi.war** will be created, which is located into the target folder. At this time, we have created the two necessary war files. Furthermore, you should move the **fiwareprivacyrestapi.war** into the **webapps** folder of the one tomcat who listens to the port **8081** and the file **uprove-web-service-0.0.1-SNAPSHOT.war**, which is located into the **ABAC-Uprove/eu.recred.uprove/uprove-web-service/target**, into the **webapps** folder to the other tomcat in port **8080**. Furthermore, the last instruction is about some changes which should be done in the source code of the application and being more specific these changes are about the IP address which is used. The used IP addresses should be changed into your local IP address and this should be applied to all of the demonstrated IP addresses.

After the above changes you will be ready to build the application and create the apk. If there is any maven packet which is not downloaded from the build, then you can manually add it to the code because it is important for the right apk's creation to find it from the following link [40]. To build the program you should just press the build button in the task bar. To create the apk file you should follow the instruction **Build → Build apk**, after the apk's creation you should import it to a device as we saw above. After this you should import it into a device, physical or virtual, as we demonstrated above. The result which you should receive is demonstrated into the following figure, Figure 5.5.

### 5.3.3 ReCRED Mobile Application

The integration of the above applications construct the ReCRED Application. With this mobile application the user is able to use the ReCRED from his phone. At this subsection we will analyze the application's architecture.

There are three participants in this architecture:

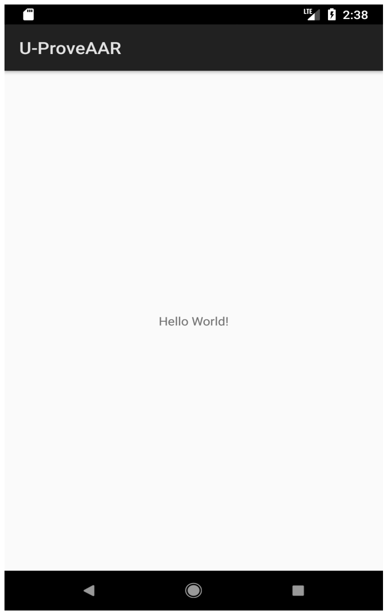


Figure 5.5: U-ProveAAR

- The user device
- The Credential manager
- The Consolidator

At first the user logs into the CM module application front end and the application retrieves the user's information from the Storage API Server. After this, the user is able to choose the credential that she desires to be issued. Once she selects the "Issue" button for a specific credential, the CM Module Application contacts the ReCRED API Server to obtain the session token for the issuance of the chosen credential. When the CM Module Application receives the session token from the API Server, then it provides the user with a 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 to the API Server. The credential which will be issued will be stored into the user's device.

The Credential Manger Module front end is a web application that can be used by the user to issue her credential by the IDC. The application is linked to the Storage API, in order to retrieve the attributes to be used in the credential generation. Also, there is a link to the ReCRED API Server, fo the actual credential issuance.

The ReCRED API Server is the server which will run the cryptographic functionalities for the IdeMix and the U-prove protocol. The ReCRED API Server contains two API Servers, the IdeMix API Server and the U-Prove API Server. The user can choose her wanted issuing protocol, depending on the choice of the user, the corresponding API Server will be used. The below figure, Figure 5.6, demonstrated the successful communication which we proposed above.

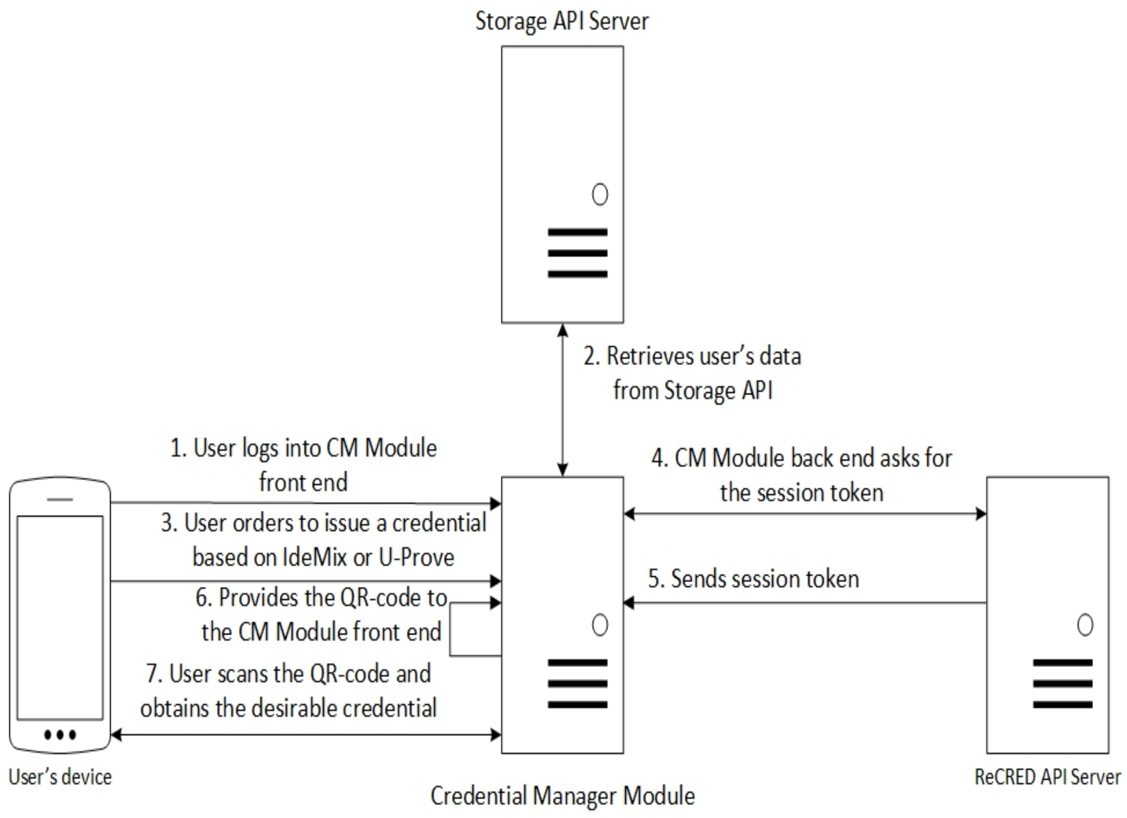

Storage API Server
2. Retrieves user's data from Storage API
1. User logs into CM Module front end
4. CM Module back end asks for the session token
3. User orders to issue a credential based on IdeMix or U-Prove
5. Sends session token
6. Provides the QR-code to the CM Module front end
7. User scans the QR-code and obtains the desirable credential
User's device
Credential Manager Module
ReCRED API Server

Figure 5.6: ReCRED Application

# Chapter 6

# Use Cases

The best way to understand how the ACS will be used in real life, is to create and demonstrate a use case. So, we will describe and demonstrate a use case for the age verification. To do this we will use the ReCRED system and its components to set up this example and we will explain exactly the steps needed from the user's side and from the SP's side to use and support this function which is called Age Gate and is a ReCRED's component.

In our life it is known that there are many reasons for the age verification because there are many websites and real life areas where there is an age restriction, for example there are websites where people who are not over eighteen (18) years old are not allowed to enter or in real life there are shops which sell alcohol and tobacco but the customers of these products have to be over eighteen (18) years old, but the administrators of these systems do not want to see the whole identity of the customers, they just want to be sure that they are legal users. For these real-life examples, the current verification system reveals more personal data than the expected user's age which is the only required information. But, the ReCRED which implemented ACS based on the IdeMix provides a very different verification way which will only be applied if the user meets the requirements. In the following paragraphs we will analyze the use case exactly.

## 6.1 User's Registration

At first the user, who wants to have access to the age restricted websites, must register with an IDP supported by ReCRED and then verify her age. So, the user registers with the ReCRED, through the Authentication Management Module which also provides her fingerprint to the FIDO Server. After that the user can use the Credential Management module to issue the cryptographic credential which will prove her age to the SPs and can securely store it in her mobile device. The following figure, Figure 6.1, demonstrates exactly how a user will register to the system, where the continued arrows are the actions and the figures are the participants of each actions.

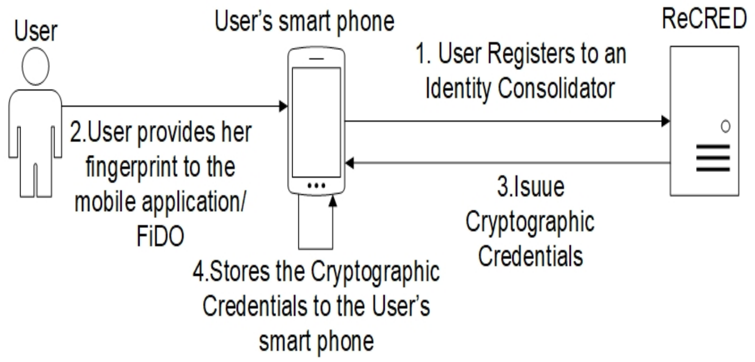

User
User's smart phone
ReCRED
1. User Registers to an Identity Consolidator
2.User provides her fingerprint to the mobile application/ FiDO
3.Isuue Cryptographic Credentials
4.Stores the Cryptographic Credentials to the User's smart phone

Figure 6.1: End-user registration

## 6.2 SP's Registration

Moreover, the SP must register to the Age Gate of ReCRED to be able to provide this functionality. At first the SP registers with the Age gate module, by filling in a simple form. Then the SP can register the wanted websites and for each website she must define the website's title, a short description, the URL and the age restriction policy i.e. the user has to be over 18 years old. After this submission form the side of SP, an Age Gate operator will review each request and will approve them. Also, after the approval a script will be sent to the SP, an OIDC Client, [41], who will add it to the website. When the SP adds the new script to the website then the new visitors will be able to use the Age Gate solution to verify their age to the website. The following figure, Figure 6.2, demonstrates exactly how a SP will register to the system of the Age Gate module, where the continued arrows are the actions and the figures are the participants of each actions. The shortcut of AGP means Age Gate Platform.

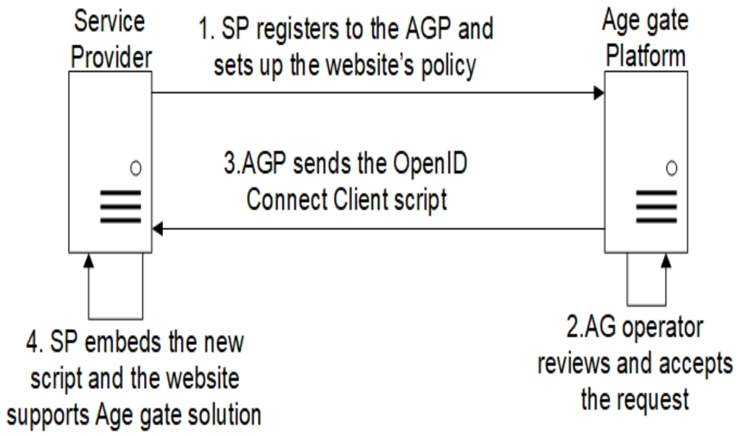

Service Provider
1. SP registers to the AGP and sets up the website's policy
Age gate Platform
3.AGP sends the OpenID Connect Client script
4. SP embeds the new script and the website supports Age gate solution
2.AG operator reviews and accepts the request

Figure 6.2: Website Registration

## 6.3 Use Cases

Furthermore, the proof of age can be done in two ways, through an age cryptographic credential which is stored into the user's mobile device or through the OIDC. In the below subsections we will analyze the two use cases.

### 6.3.1 First Use Case

The user tries to visit the age-restricted website using his computer. Then the website asks from the Age Gate Server to verify the user's age. The Age Gate Server returns a QR-code, which is displayed in the age-restricted website and the user can use her Age Gate Mobile Application to scan the QR-code and prove her age. However this action is not required if the user tries to connect to the website through her mobile device. The Age Gate mobile application will search for cryptographic credential stored in the mobile device which will prove the user's age and will send it to the Age Gate Server. After this action, the Age Gate Server will be able to verify the user's age and if the user's age meets the requirements of the website's age restriction policy then the user will be able to use the website. The following figure, Figure 6.3, demonstrates exactly how a user will reach an age restricted website, where the continued arrows are the actions and the figures are the participants of each actions.

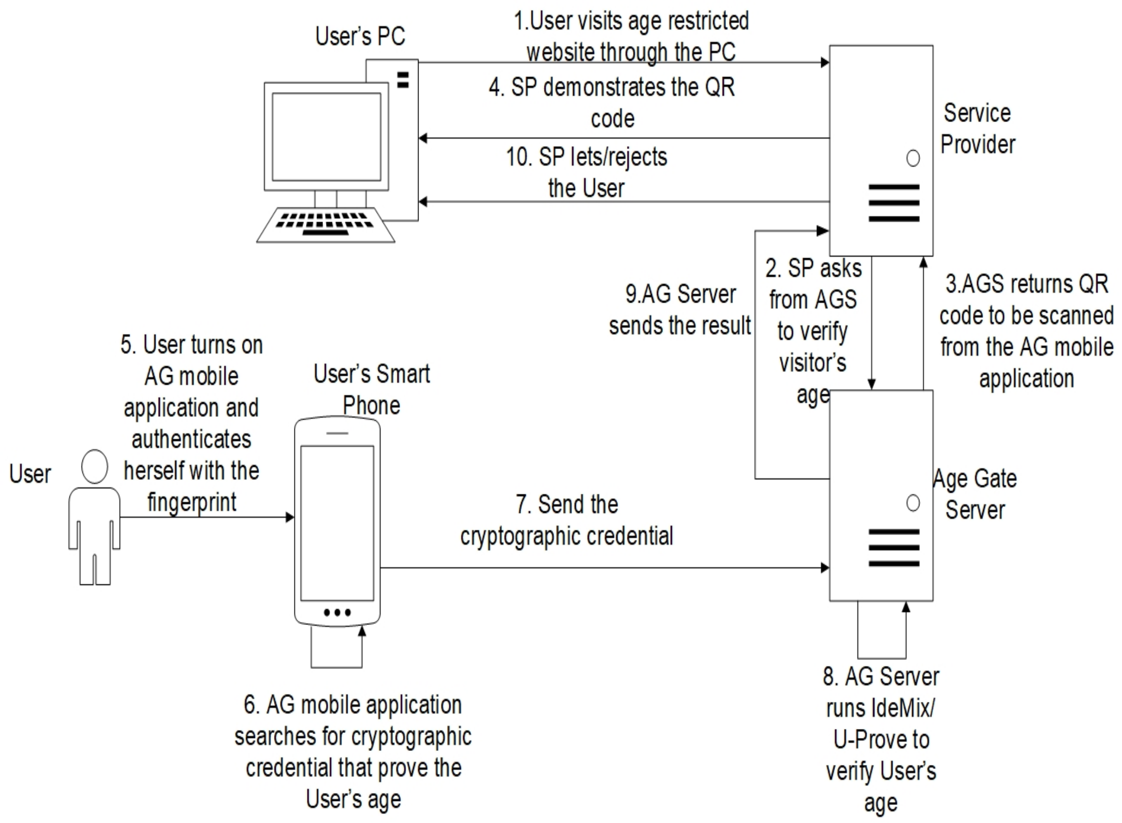

User's PC
1.User visits age restricted website through the PC
4. SP demonstrates the QR code
10. SP lets/rejects the User
Service Provider
9.AG Server sends the result
2. SP asks from AGS to verify visitor's age
3.AGS returns QR code to be scanned from the AG mobile application
5. User turns on AG mobile application and authenticates herself with the fingerprint
User's Smart Phone
User
Age Gate Server
7. Send the cryptographic credential
6. AG mobile application searches for cryptographic credential that prove the User's age
8. AG Server runs IdeMix/ U-Prove to verify User's age

Figure 6.3: Age verification

### 6.3.2 Second Use Case

In this paragraph we will, explain the Age Verification through the OIDC, [41]. Initially, the user tries to visit the age-restricted website using his computer. Then the website asks from the Age Gate Server (AGS) to verify the user's age. The AGS returns a QR-code, which is displayed in the age-restricted website and the user can use her Age Gate Mobile Application (AGMA) to scan the QR-code and prove her age, however this action is not required if the user tries to connect to the website through her mobile device. The AGMA will display all the supported IDPs and then the user will choose to prove her age through the ReCRED IDC, where the IDC is the default IDP for this use case. The AGMA enters the FIDO UAF server sitting on the IDC and then the user will provide her fingerprint to the FIDO UAF Client. The FIDO UAF strong authentication framework provides high security features of end user devices for strong authentication and reduces problems associated with the creation and reminds of the online credentials.

If the authentication is successful, then the mobile application will return an authentication token which will be sent to the AGS and then the AGS will send it to the IDC's OpenAM and request the user's age. Where the OpenAM is a web-based open-source solution that provides authentication, authorization, entitlement, and federation Services. OpenAM provides many authentication methods and offers the flexibility to create custom authentication modules based on the JAAS (Java Authentication and Authorization Service) open standard. In addition to that, OpenAM's federation Services allow federated members to securely share identity information with each other. The OpenAM will return the user's age to the AGS if the user gives her consent. Then the AGS will evaluate the age policy against the user's age and return the results.

Also, the AGMA provides a list with all the age restricted websites where the user has been granted access. The user can see detailed report with access attempts, including specific time-stamps and attempts' results and the user can revoke access to a certain age-restricted website.

#### 6.3.2.1 Architecture

At this subsection we will analyze the special architecture which is used in this use case because of the OIDC's usage. OIDC contains three participants, the Idenity Provider, the Relying Party and the user. However, the participant that we will analyze is the IDC who acts as an OIDC provider. The IDC acts as a trusted OIDC provider and holds the verified age of the end-user. Also, it provides mechanisms to end-users in order to regisred and prove their age. Furthermore, IDC includes FIDO server, in order to be able to perform FIDO UAF authentication. It also includes an attributes database that maps the user's idenity with confirmed identity attributes, so the IDC can verify the user's age. IDC contains the gateSAFE module which provides single-sign-on implementation. With this feature the IDC acts as OIDC provider and provides authentication to the AGS. The other participants contain the requisite daemons to be compatible with the OIDC protocol

# Bibliography

[1] Identity Mixer, `https://idemix.wordpress.com/2009/08/18/quick-intro-to-credentials/`

[2] Gregory Neven, *A Quick Introduction to Anonymous Credentials, IBM Zrich Research Laboratory, August 2008.*

[3] Foteini Baldimtsi and Anna Lysyanskaya, *Anonymous credentials light.* In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (CCS '13). ACM, New York, NY, USA, 1087-1098.

[4] S. Goldwasser, S. Micali, and C. Rackoff *The knowledge complexity of interactive proof systems. SIAM J. Comput. 18, 1 (February 1989), 186-208p.*

[5] ] Fiat A., Shamir A. , *How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko A.M. (eds) Advances in Cryptology CRYPTO' 86. CRYPTO 1986. Lecture Notes in Computer Science, vol 263. Springer, Berlin, Heidelberg.*

[6] U. Feige, A. Fiat, and A. Shamir, . *Zero knowledge proofs of identity. In Proceedings of the nineteenth annual ACM symposium on Theory of computing (STOC '87), Alfred V. Aho (Ed.). ACM, New York, NY, USA, 210-217p.*

[7] Jean-Jacques Quisquater, Louis Guillou, Marie Annick, and Tom Berson, *How to explain zero-knowledge protocols to your children. In Proceedings on Advances in cryptology (CRYPTO '89), Gilles Brassard (Ed.). Springer-Verlag New York, Inc., New York, NY, USA, 628-631p.*

[8] Manuel Blum, Paul Feldman, and Silvio Micali, *Non-interactive zero-knowledge and its applications. In Proceedings of the twentieth annual ACM symposium on Theory of computing (STOC '88). ACM, New York, NY, USA, 103-112.*

[9] Claus-Peter Schnorr,*Efficient Identification and Signatures for Smart Cards. In: Brassard G. (eds) Advances in Cryptology CRYPTO' 89 Proceedings. CRYPTO 1989. Lecture Notes in Computer Science, vol 435. Springer, New York, N.*

[10] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, Stefan Wolf, *Pseudonym Systems . in the Sixth Annual Workshop on Selected Areas in Cryptography (SAC '99). 1999: Springer-Verlag LNCS.*

[11] David Chaum, *Blind Signatures for Untraceable Payments. In: Chaum D., Rivest R.L., Sherman A.T. (eds) Advances in Cryptology. Springer, Boston, MA.*

[12] David Chaum, *Security without identification: transaction systems to make big brother obsolete. Commun. ACM 28, 10 (October 1985), 1030-1044p.*

[13] Camenisch Jan, Lysyanskaya Anna, *An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In: Pfitzmann B. (eds) Advances in Cryptology EUROCRYPT 2001. EUROCRYPT 2001. Lecture Notes in Computer Science, vol 2045. Springer, Berlin, Heidelberg.*

[14] Jan Camenisch, Els Van Herreweghen, *Design and implementation of the idemix anonymous credential system. In Proceedings of the 9th ACM conference on Computer and communications security (CCS '02), Vijay Atluri (Ed.). ACM, New York, NY, USA, 21-30p.*

[15] ] Jan Camenisch, Anna Lysyanskaya, *A Signature Scheme with Efficient Protocols. In: Cimato S., Persiano G., Galdi C. (eds) Security in Communication Networks. SCN 2002. Lecture Notes in Computer Science, vol 2576. Springer, Berlin, Heidelberg.*

[16] ] Jan Camenisch and Els Van Herreweghen, *Design and implementation of the idemix anonymous credential system. In Proceedings of the 9th*

*ACM conference on Computer and communications security (CCS '02), Vijay Atluri (Ed.). ACM, New York, NY, USA, 21-30p.*

[17] IRMA, `https://www.irmacard.org/irma/`

[18] Patrik Bichsel, Jan Camenisch, Thomas Gro, and Victor Shoup, *Anonymous credentials on a standard java card. In Proceedings of the 16th ACM conference on Computer and communications security (CCS '09). ACM, New York, NY, USA, 600-610.*

[19] Sabouri A., Krontiris I., Rannenberg K.,Attribute-Based Credentials for Trust (ABC4Trust). In: Fischer-Hbner S., Katsikas S., Quirchmayr G. (eds) Trust, Privacy and Security in Digital Business. TrustBus 2012. Lecture Notes in Computer Science, vol 7449. Springer, Berlin, Heidelberg.

[20] IBM Research - Zurich,*Specification of the Identity Mixer Cryptographic Library Version 2.3.0**

[21] R. Bhaskar, K. Chandrasekaran, S.V. Lokam, P.L. Montgomery, R. Venkatesan, Y. Yacobi, *Vulnerabilities in Anonymous Credential Systems, In Electronic Notes in Theoretical Computer Science, Volume 197, Issue 2, 2008, Pages 141-148, ISSN 1571-0661.*

[22] A. Damodaram, H. Jayasri, *Authentication without Identification Using Anonymous Credential System, IJCSIS, 2009.*

[23] Chaum D., Evertse JH, *A Secure and Privacy-Protecting Protocol for Transmitting Personal Information Between Organizations. In: Odlyzko A.M. (eds) Advances in Cryptology CRYPTO' 86. CRYPTO 1986. Lecture Notes in Computer Science, vol 263. Springer, Berlin, Heidelberg.*

[24] Chen L.,*Access with pseudonyms. In: Dawson E., Goli J. (eds) Cryptography: Policy and Algorithms. Lecture Notes in Computer Science, vol 1029. Springer, Berlin, Heidelberg.*

[25] ReCRED:`https://www.recred.eu`

[26] Identity Mixer Overview: `http://www.research.ibm.com/labs/zurich/idemix`

[27] FIDO:https://fidoalliance.org

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

[29] Pydemix Implementation:https://github.com/netgroup/pydemix

[30] Christian Paquin, Greg Zaverucha ,*U-Prove Cryptographic Specification V1.1 Revision 3.*

[31] IRMA(I Reveal My Attributes) Project:https://www.irmacard.org/

[32] IRMA:IdeMix implementation targeted on mobile devices:https://github.com/credentials/credentials_idemix

[33] FIWARE:https://www.fiware.org/

[34] ABC4trust:https://abc4trust.eu/

[35] GitHub ZHAW's Privacy Geri in FIWARE: https://github.com/Fiware/security.P2abcengine

[36] p2abcengine:https://github.com/p2abcengine/p2abcengine

[37] Ubuntu Download:https://www.ubuntu.com/download/desktop

[38] Android Studio Download:https://developer.android.com/studio/index.html

[39] Tomcat Download:http://tomcat.apache.org/

[40] Maven Download:https://mvnrepository.com/

[41] Open ID Connect:https://openid.net/connect/