DOCTORAL PROGRAMME IN
INFORMATION AND COMMUNICATION TECHNOLOGY

Doctoral candidate

Giada Sciarretta

Cycle	**30°**
Thesis	**A Methodology for the Design and Security Assessment of Mobile Identity Management: Applications to real-world scenarios**
Advisor	**Silvio Ranise (Fondazione Bruno Kessler)**
Co-advisor	**Roberto Carbone (Fondazione Bruno Kessler)**
Co-advisor	**Alessandro Armando (Università di Genova)**

1. List of publications

- G. Sciarretta and A. Armando and R. Carbone and S. Ranise. *Security of Mobile Single Sign-On: A Rational Reconstruction of Facebook Login Solution*. Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 4: SECRYPT, pages 147-158. https://doi.org/10.5220/0005969001470158.
- G. Sciarretta, R. Carbone and S. Ranise. *A delegated authorization solution for smart-city mobile applications*. Proceedings of the 2nd International Forum on Research and Technologies for Society and Industry leveraging a better tomorrow (RTSI 2016). https://doi.org/10.1109/RTSI.2016.7740623.
- G. Sciarretta, R. Carbone, S. Ranise and A. Armando. *Anatomy of the Facebook solution for mobile single sign-on: Security assessment and improvements*. Journal of Computers & Security (COSE 2017). https://doi.org/10.1016/j.cose.2017.04.011.
- G. Sciarretta, R. Carbone, S. Ranise and L. Viganò. *Design, Formal Specification and Analysis of Multi-Factor Authentication Solutions with a Single Sign-On Experience*. Proceedings of the 7th International Conference on Principles of Security and Trust (POST 2018). https://doi.org/10.1007/978-3-319-89722-6_8.

2. Research/study activities

- Participation in research projects:
    - *FIDES* (acronym for Federated IDEntity System) was a two-year activity of the Privacy, Security and Trust action line of EIT Digital with the goal of enabling the secure access of digital and physical services using the users' digital identity in a federated and cross-border ecosystem.
    - *TreC* (acronym for Cartella Clinica del Cittadino, i.e. Citizens' Clinical Record): we have proposed a solution that allow patients to access different TreC e-health mobile applications (and possibly other third-party e-health applications) through a single authentication act.
    - *DigiMat-Lab* is a joint laboratory between FBK and IPZS (acronym for “Istituto Poligrafico e Zecca dello Stato"), which is the Italian state printing office and mint. We are involved in various activities, one is the design of a mobile multi-factor authentication mechanism that uses the Italian electronic identity card (CIE 3.0 - Carta d'Identità Elettronica) as an OTP generator.
- Conferences and Workshops as speaker:
    - OSW16 (OAuth Security Workshop, 2016) with the talk “*An OAuth-based Single Sign-On solution for Mobile Applications*”

    - ITA-SEC17 (Italian Conference on Cybersecurity, 2017) with the talk "*An OAuth-based Single Sign-On Solution for Mobile Applications*"
    - ITA-SEC18 (Italian Conference on Cybersecurity, 2018) with the tutorial *"Secure and Usable Mobile Identity Management Solutions: a Methodology for their Design and Assessment"*
    - OSW18 (OAuth Security Workshop, 2018) with the talk "*Experiences Using OAuth 2.0 in Federated and Multichannel Open Service Platform*"

- Traineeship (Erasmus+ experience) of 3 months at the King's College of London with the title "Formal analysis of authentication solutions for native applications"
- "BEST Innovation Idea AWARD" at the SECENTIS PhD Symposium (DBSec 2016)
- Program Committee Member for the 3rd OAuth Security Workshop (OSW18)

UNIVERSITÀ DEGLI STUDI
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
ICT International Doctoral School

A Methodology for the Design and Security Assessment of Mobile Identity Management:

Applications to real-world scenarios

Giada Sciarretta

Advisor

Silvio Ranise, Fondazione Bruno Kessler

Co-Advisors

Alessandro Armando, Università degli Studi di Genova (DIBRIS)

Roberto Carbone, Fondazione Bruno Kessler

May 2018

Abstract

The widespread use of digital identities in our everyday life, along with the release of sensitive data on many online transactions, calls for Identity Management (IdM) solutions that are secure, privacy-aware, and compatible with new technologies, such as mobile and cloud computing. While there exist many secure IdM solutions for web applications, their adaptation in the mobile context is a new and open challenge. The majority of mobile IdM solutions currently used are based on proprietary protocols and their security analysis lacks standardization in the structure, definitions of notions and entities, and specific considerations to identify the attack surface that turns out to be quite different from well understood web scenarios. This makes a comparison among different solutions very complex or, in the worst case, misleading. To overcome these difficulties, we propose a novel methodology for the design and security assessment of mobile IdM solutions. The design space is characterized by the identification of: (i) national (e.g., SPID for Italy) and European (e.g., eIDAS) laws, regulations and guideline principles that are particularly relevant to digital identity and privacy; (ii) a list of security and usability requirements that are related to IdM solutions (e.g., single sign-on and multi-factor authentication); (iii) a set of implementation mechanisms that are relevant to authentication and authorization on mobile devices and simplify the satisfaction of the requirements in (ii). All the designed solutions use as blueprint a reference model resulting from a rational reconstruction of the mobile IdM solution adopted by Facebook and a study of the OAuth specification for native applications. Regarding the security assessment, our methodology supports analyses ranging from semi-formal to formal. For the former, an IdM

designer is required to specify the security relevant parts of the protocol using message sequence charts, the threat model and the security properties; these offer the starting point to argue whether the protocol satisfies the specified properties. For the latter, an IdM designer is required to specify the protocol flow, the attacker properties and the security properties using one of the available formal specification languages for the description of cryptographic and browser-based protocols, and verify the security property violations using an automated tool for protocol analysis. To validate our approach, we applied it to four different real-world scenarios that represent different functional and usability requirements:

1. *TreC: a multi-factor authentication solution with a single sign-on experience for mobile e-Health applications.*
2. *Smart Community: a secure delegated access solution in the context of smart-cities.*
3. *DigiMat-Lab (Istituto Poligrafico e Zecca dello Stato): a mobile multi-factor authentication solution that uses as second factor the Italian electronic identity card.*
4. *FIDES: an IdM solution that combines federation and cross-border aspects in the context of the European single digital market.*

The custom designs obtained by applying our methodology in the four scenarios above show the generality and effectiveness of our methodology. When using formal analysis, we have re-used the specification language and tools developed in the context of the AVANTSSAR EU-founded project.

Keywords

Digital Identity, Authentication and Authorization, Mobile Devices, Single Sign-On, OAuth 2.0.

Contents

List of Tables

List of Figures

Abbreviations

AA	Activation Assumption
ADC	Authenticazione Del Cittadino (Citizen's Authentication)
AP	Authorization Provider
AppCtx	Application Context
AS	Authorization Server
BA	Background Assumption
C	Client
CA	Communication Assumption
CIE	Carta d'Identità Elettronica (Italian Electronic Card)
CR	Challenge Response
FB	Facebook
HWToken	Hardware Token
IdB	Identity Broker
IdM	Identity Management
IdP	Identity Provider
LoA	Level of Assurance
MFA	Multi-Factor Authentication
MSC	Message Sequence Chart
OIDC	OpenID Connect
OTP	One-Time Password
PA	Public Administration

PAL Protocol Assurance Level
PHR Personal Health Record
RO Resource Owner
RS Resource Server
SC Smart Community
SDK Software Development Kit
SP Service Provider
SSO Single Sign-On
TA Trust Assumption
TOTP Time-based OTP
TP Token Provider
TreC Cartella Clinica del Cittadino (Citizens' Clinical Record)
UA User Agent
UBA User Behavior Assumption

Chapter 1

Introduction

We use our digital identities everyday, from accessing our email account to online shopping. Underlying these transactions, there are Identity Management (IdM) protocols that exchange user's attributes among the different entities involved in the communication. An analysis of the data breaches occurred in the first half of 2017 [58] shows that identity theft accounted for 74% of all data breaches in this time range — there were 918 reported data breaches and almost 1.9 billion of compromised data. These numbers show how it becomes extremely important to design solutions that augment the security of IdM systems. In this context, a central theme in the European Union is the definition of a common regulation on digital identity (e.g., eIDAS [50]) that guides both public and private sectors to define the security requirements and mitigate the known attacks and vulnerabilities. Many EU member states have also defined new national regulations (e.g., SPID [9] in Italy) and collaborate to many digital identity initiatives (e.g., STORK 2.0 [22]).

In general, IdM refers to different aspects of the digital identity lifecycle (e.g., the creation and provision of identities, password management, authentication, and so on). In this study, we use IdM to indicate the aspects related to authentication and authorization. According to RFC 4949 [89],

authorization is defined as "the process of granting approval to a system entity to access a system resource." While authentication is "the process of verifying a claim that a system entity or system resource has a certain attribute value. An authentication process consists of two steps: an *identification step*, presenting the claimed attribute value (e.g., a user identifier) to the authentication subsystem, and a *verification step*, presenting or generating authentication information (e.g., a value signed with a private key) that acts as evidence to prove the binding between the attribute and that for which it is claimed." It is important to notice that authentication is closely related to authorization (e.g., authenticated identities are the basis for access control).

The design of a secure authentication and authorization solution should take into consideration also usability aspects as either difficult to use solutions will not be used at all or people will develop shortcuts that may severally reduce security (e.g., using guessable passwords). Thus, the highest level of security can be obtained only with an equally high level of usability. An example of usable feature is the support of a Single Sign-On (SSO) experience: users have to enter their credentials only once and then can access different services.

In this work, our focus is on mobile applications. While for web applications there exist many secure authentication and authorization solutions to protect the user's digital identity and online resources, solutions for the mobile context are not well consolidated. This is a huge limitation as mobile applications are dominating the market of online services, with a total of 5.1 billion unique mobile subscribers by the end of 2017 [70] (more than 65% of the world population). In addition, during the last year we have observed a growth of 50% in Internet usage on mobile devices, in contrast to a decrease of 20% in Internet usage on laptops and desktops [110].

It is important to note that people use mobile devices not only to access

social networking applications and games, but also security-critical applications (e.g., e-banking and e-health). Thus, different levels of assurance must be taken into consideration designing an IdM solution: from a basic authentication with a self-declared identity to a Multi-Factor Authentication (MFA) — that augments the security of a basic authentication combining two or more authentication factors (e.g., a password combined with a biometric data) — with a strong proof of the user's real world identity.

In the rest of the chapter, we describe the current security issues and open challenges in this area (Sect. 1.1). We provide a first glance at our methodology and contributions (Sect. 1.2). Finally, the overall structure of the thesis is presented (Sect. 1.3).

1.1 Problem Statement

Existing standards (e.g., OAuth 2.0 [64] and OpenID Connect [94]) provide only a partial support for mobile native applications, leaving many implementation choices to developers. Leveraging these protocols, many social networks have already deployed their own authentication and authorization solutions, which have been tremendously successful: most Facebook and Google users routinely and transparently use them on their smartphones and tablets. However, the adaptation of protocols originally designed to work in a traditional web scenario have caused the spread of a number of serious vulnerabilities and attacks. A best practice for authentication and authorization for mobile applications has been released by the OAuth working group with the title "OAuth 2.0 for Native Apps" [93]. However, there are a number of aspects which — to the best of our knowledge — have not yet been satisfactorily addressed.

Regarding MFA, we have analyzed solutions that use *One Time Passwords (OTPs)*, which are passwords that are valid for a short time and

can only be used once. We have selected OTP generation algorithms as they are commonly used to provide strong authentication and many alternative solutions (from physical to software tools) are available on the market. For instance, Google Authenticator is a mobile app that generates OTPs [60]. Like Google Authenticator, many of the OTP solutions on the market can be readily applied only to web solutions where mobile devices are used as an additional factor. However, as explained before, the number of users accessing web apps on their laptops is decreasing in favor of an exponential increase of mobile apps, and — to the best of our knowledge — the definition of a MFA solution for native apps is still not well specified. Even if there are some solutions currently used, their security analyses have been performed informally or semi-formally at best, and without following a standard procedure. This makes a comparison between the different solutions both complex and potentially misleading.

Due to this lack of specifications and security guidelines, designing a mobile IdM solution that covers all these aspects (e.g., SSO and MFA) from scratch is not a simple task; and as its security depends on several trust and communication assumptions, in most cases, could result in a solution with hidden vulnerabilities. An example of a wrong design choice is the use of an embedded browser (i.e. a browser hosted inside an app) as user agent (UA) — entity that manages the interactions between a service provider (SP) and an identity provider (IdP). The use of this type of browser is widely discouraged, as there is a loss of isolation between SP and UA. If SP is malicious, then it can steal the user credentials or change the authorization permissions. An additional limitation when choosing an embedded browser is that it does not provide a SSO experience. Indeed, if the browser is integrated within the app, then the login session information is stored in (and only accessible to) the app and it is therefore not available to other apps. This forces the user to re-enter credentials even if she has

an active login session with an *IdP*. Another example of a wrong design choice is the use of SMSs to send OTPs to users, as the confidentiality of an SMS text is debatable. In [91], the National Institute of Standards and Technology (NIST) deprecates multi-factor solutions that use SMS because of the many vulnerabilities.

Regarding usability, solutions designed for desktop platform cannot be trivially reused. For example, reusing OTP generator apps that show OTP values on the screen is not a usable choice, as the user has to memorize the OTP and move from a browser to an app and back; this severally impairs the user experience.

These examples clearly highlight the need for a methodology covering various aspects of mobile IdM solutions.

1.2 Our Methodology and Contributions

In this work, we present a methodology for the design and security assessment of mobile IdM solutions. The design space is characterized by the identification of: *(i)* Italian and European laws, regulations and guideline principles that are particularly relevant to digital identity and privacy; *(ii)* a list of functional and usability aspects that are related to IdM solutions (e.g., SSO and MFA); *(iii)* a set of implementation mechanisms that are relevant to authentication and authorization on mobile devices. As part of the design process, we present our reference model for a solution that allows users to access different mobile apps through an authentication solution providing, at the same time, a SSO experience and MFA support. For what concerns the security assumptions and the design of our reference model, our work is based on Facebook and OAuth, which represent two different design choices:

- Starting from the Facebook solution (cf. Sect. 4.2) and taking ad-

vantage of our security analysis (cf. Sect. 6.1.3), we have derived a reference model for native IdM solutions with a native application as user agent (entity that manages the interaction between users, apps and identity providers/authorization servers).

- Starting from OAuth for native app (cf. Sect. 2.2.4), we have derived a reference model with an external browser as user agent.

For these two variants, we have detailed the flow (message exchange and security checks) and provided a formal specification.

Regarding the security assessment, our methodology supports analyses ranging from semi-formal to formal. For the former, an IdM designer has to specify the security relevant parts of the protocol using message sequence charts, the threat model and the security properties; these offer the starting point to argue whether the protocol satisfies the specified properties. For the latter, an IdM designer is required to specify the protocol flow, the attacker properties and the security properties using one of the available formal specification languages for the description of cryptographic and browser-based protocols, and verify the security property violations using an automated tool for protocol analysis. Provided that the IdM designer performs the mapping between the security properties — which we have provided in a clear natural-language description — to the selected formal language, our methodology is parametric on the formal specification language. In our formal analysis, we have re-used the specification language (ASLan++ [32]) and one of the tools (SATMC [29]) developed in the context of the AVANTSSAR EU-founded project [26]. In our methodology, we require the use of a formal analysis when the solution deals with sensitive data or the overall risk is high. Finally, our methodology provides a list of usable aspects to consider during the design and a questionnaire template for evaluating user satisfaction.

Summarizing, we have defined a methodology for the design and security assessment of mobile IdM solutions, and made the following contributions:

- *for the design:* the flow, security assumptions and goals of a reference model (called *mID(OTP)*) that provides a SSO experience and a MFA support for native apps using OTPs;
- *for the security assessment:* an adversarial model for the semi-formal approach and a formal mapping (from an informal description to ASLan++) of *mID(OTP)* flow, assumptions and goals (also for a MFA solution) for the formal approach;
- *for the usability analysis:* a set of usability considerations and a questionnaire template for evaluating user satisfaction.

To validate our methodology and the corresponding reference model, we have applied it to four different real-world scenarios that represent different functional and usability requirements.

1.3 Structure of the Thesis

The rest of the thesis is structured as follows:

Chapter 2 provides the basic notions related to IdM standards. In Sect. 2.1, we define the roles typically involved in authentication and authorization solutions for native mobile applications and discuss different OTP-generation approaches (Sect. 2.1.1). In Sect. 2.2, we describe the most widespread authentication and authorization standards and discuss the state of the art of these standards in the mobile context.

Chapter 3 describes laws and regulations that are particularly relevant to digital identity and privacy in the European and Italian context. In Sect. 3.1, we define basic concepts of IdM solutions by referring

to IdM technical standards and recommendations. Then, we identify European and Italian laws and regulations on digital identity and privacy (Sect. 3.2). In addition, for the Italian laws, we report a decisional diagram that could be used to identify which legal obligations a specific IdM solution has to follow (Sect. 3.2.3).

Chapter 4 introduces our methodology for the design and the security assessment of mobile IdM solution. In Sect. 4.1, we provide an overview of the different phases of our methodology. Then, we discuss some aspects that are essential to understand our approach. First, Sect. 4.1.1 discusses the need for a reference model that can be used as a starting point for the design of IdM solutions. Second, in Sect. 4.1.2, we present the innovative relationship between implementation and design that our methodology subsumes, and we explain the main Android implementation features that are related to the authentication and authorization aspects. Then, to complete the overview, we discuss the relationships between the identified features of Android and iOS. Finally, in Sect. 4.2 we present our rational reconstruction of Facebook social login for Android native apps that is at the basis of our reference model.

Chapter 5 describes the first two phases of our methodology. Sect. 5.1 describes the first phase of our methodology, where the IdM designer is required to fill the Application Context (*AppCtx*) for the specific application scenarios. Sect. 5.2 describes the second phase of our methodology and details our reference model, called *mID(OTP)*, for mobile IdM solutions. In Sect. 5.2.1, we describe *mID(OTP)* flow for two variants: *mID(OTP) Var1* with a native application as *UA* based on the Facebook solution and *mID(OTP) Var2* with an external browser as *UA* based on OAuth. In Sect. 5.2.2, we identify the

security assumptions for *mID(OTP)* in both variants (native app or browser) and argue that they allow to prevent a violation of the security goals that we have identified in Sect. 5.2.3. Finally, in Sect. 5.2.4, we compare *mID(OTP)* with Facebook and OAuth for native apps solutions.

Chapter 6 describes the third and last phase of our methodology. Regarding the security analysis, we have considered two levels: semi-formal (described in Sect. 6.1) and formal (described in Sect. 6.2). For the former, in Sect. 6.1.3, we provide an example for the Facebook solution. For the latter, in Sect. 6.2.1, we describe how the semi-formal description of the *mID(OTP)* solution can be translated into a formal model (in this case, specified in ASLan++). Regarding the usability analysis, Sect. 6.3 details our approach.

Chapter 7 applies the *mID(OTP)* reference model to four real use-case scenarios: mobile electronic health (project called *TreC* - Sect. 7.1), smart city and secure delegated access (project called *Smart Community* - Sect. 7.2), national electronic identity (project called *DigiMatLab* - Sect. 7.3), and digital single market with multi IdPs (EIT activity called *FIDES* - Sect. 7.4).

Chapter 8 discusses the most relevant work related to IdM. Sect. 8.1 discusses papers whose focus is the security analysis of OAuth 2.0 and OpenID Connect. In Sect. 8.2, we illustrate some of the current European initiatives on digital identities. Finally, in Sect. 8.3, we present emerging IdM technologies on the market.

Chapter 9 draws conclusions and discusses future work.

Some of the material in this manuscript is based on the following journal and papers published at international conferences:

- G. Sciarretta and A. Armando and R. Carbone and S. Ranise. *Security of Mobile Single Sign-On: A Rational Reconstruction of Facebook Login Solution*. Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 4: SECRYPT, pages 147–158. `https://doi.org/10.5220/0005969001470158`.

- G. Sciarretta, R. Carbone and S. Ranise. *A delegated authorization solution for smart-city mobile applications*. Proceedings of the 2nd International Forum on Research and Technologies for Society and Industry leveraging a better tomorrow (RTSI 2016). `https://doi.org/10.1109/RTSI.2016.7740623`.

- G. Sciarretta, R. Carbone, S. Ranise and A. Armando. *Anatomy of the Facebook solution for mobile single sign-on: Security assessment and improvements*. Journal of Computers & Security (COSE 2017), F71, pages 71-86 (2017). `https://doi.org/10.1016/j.cose.2017.04.011`.

- G. Sciarretta, R. Carbone, S. Ranise and L. Viganò. *Design, Formal Specification and Analysis of Multi-Factor Authentication Solutions with a Single Sign-On Experience*. Proceedings of the 7th International Conference on Principles of Security and Trust (POST 2018). `https://doi.org/10.1007/978-3-319-89722-6_8`.

Chapter 2

Background on IdM Standards

This chapter provides the basic notions related to IdM standards. In Sect. 2.1, we define the roles typically involved in authentication and authorization solutions for native mobile applications and discuss different OTP-generation approaches (Sect. 2.1.1). In Sect. 2.2, we describe the most widespread authentication and authorization standards and discuss the state of the art of these standards in the mobile context.

2.1 Authn and Authz: Roles and Flow

As described in Chapter 1, in general, IdM refers to different aspects of the digital identity lifecycle (e.g., the creation and provision of identities, password management, and so on). We use IdM to indicate the aspects related to authentication and authorization.

An authentication protocol determines the validity of one or more authentication factors used to claim a digital identity of an entity. In our analysis, we will focus on both username-password authentication and multi-factor authentication solutions. A multi-factor authentication solution augments the security of the basic username-password authentication by exploiting two or more authentication factors. In [46], it is defined as:

> *"a procedure based on the use of two or more of the following elements — categorised as knowledge, ownership and inherence: i) something only the user knows, e.g., static password, code, personal identification number; ii) something only the user possesses, e.g., token, smart card, mobile phone; iii) something the user is, e.g., biometric characteristic, such as a fingerprint. In addition, the elements selected must be mutually independent [...] at least one of the elements should be non-reusable and non-replicable".*

The more factors are used during the authentication process, the more confidence a service has that the user is correctly identified.

An authorization protocol regulates what an entity is allowed to do. In our analysis, we focus on *access delegation*, which is a mechanism by which the data owner can delegate access to a selected subset of the data to a designated application, without enabling the application to impersonate the data owner. The mechanism is composed of two steps: one in which applications can request access to selected pieces of information and one in which users instruct servers to grant (or deny) requested accesses. The access delegation usually results in the release of an *access token* for the application. An access token can be seen as a capability, i.e. as a communicable and unforgeable token containing a reference to a resource together with a list of permissions. Compared to an Access Control List (ACL) system where each attempt of accessing a resource is verified checking the corresponding ACL, in a capability system, by simply presenting the capability to a resource, a subject can perform an action on the resource — provided that this action is in the list of permissions of the capability.

Many standards and products on authentication and authorization have been developed in the last few years. In the context of native mobile applications, they typically involve entities who play these roles: a *User* (*User*) that wants to access a native application; a *Service Provider* (*SP*) that has

a client side (SP_{app}) — that corresponds to the native application *User* wants to access — and (optionally) a server side (SP_S); the server-side of an *Identity Provider* (IdP_S) that manages *User* authentication; a *User Agent* (UA), which could be either a browser or a native app used to perform the authentication process between SP_{app} and IdP_S. Optionally, in case IdP_S does not have all the information on the *User* required by SP_{app}, an *Attribute Provider* (AP) is invoked. In some scenarios, an identity federation can be achieved thanks to the interaction with an *Identity Broker* (IdB) that mediates the exchange of digital identities between SP_{app} and IdP_S.

In the case of multi-factor authentication, the authentication process usually involves also physical devices or software for the generation of additional identity proofs. We indicate with *Token Provider* (TP), the role that manages the generation and validation of additional identity proofs — that may have a server (TP_S) and a client side (TP_{app}) — and with *Hardware Token* ($HWToken$), a physical device used in combination with the user smartphone (e.g., smartcard and USB keys).

In the case of authorization, we have to consider also the *Authorization Server* (AS) that manages the release of access tokens to SP, and the *Resource Server* (RS) where the resources are stored.

Combining all the scenarios, we define the following set of IdM roles (cf. Table 2.1 for their extended names):

$$IdMRoles = \{User,\ SP_{app},\ SP_S,\ UA,\ IdP_S,\ AP,\ IdB,\ TP_{app},\ TP_S,\ HWToken,\ AS,\ RS\}$$

The entities playing these roles could belong to the same security domain or not, if they are of different domains, they can collaborate through a federation. Often an entity can play different roles.

The simplest scenario involves three entities and five roles: a person (playing the role of *User*) who wants to authenticate in an application

(playing the role of SP_{app} and UA) using her SP credentials with SP_S (playing the role of SP_S and IdP_S). Figure 2.1 shows a more complex example that combines authentication and authorization where: a native app plays the role of SP_{app}; its server plays the role of SP_S, AS and RS; a browser plays the role of UA; and an entity (e.g., Google) plays the role of IdP_S and AP.

The flow between these entities is composed of the following steps:

1. $User$ wants to access a RS resource using SP_{app} on her mobile device;

2. In order to grant or deny the access, SP_{app} needs to know the user identity, so it will ask IdP_S, which specifically manages user identities, for this service;

3.-4. IdP_S authenticates $User$ and sends the authentication response to SP_{app}, which through UA redirects the response to SP_S;

5. SP_S asks other information about $User$ to AP;

Table 2.1: Acronyms and Extended Names of the IdM Roles.

Acronym	**Extended Name**
$User$	User
SP_{app}	Service Provider client app
SP_S	Service Provider server side
UA	User Agent (browser or native app)
IdP_S	Identity Provider server side
AP	Attribute Provider
IdB	Identity Broker
TP_{app}	Token Provider client app
TP_S	Token Provider server side
$HWToken$	Hardware Token
AS	Authorization Server
RS	Resource Server

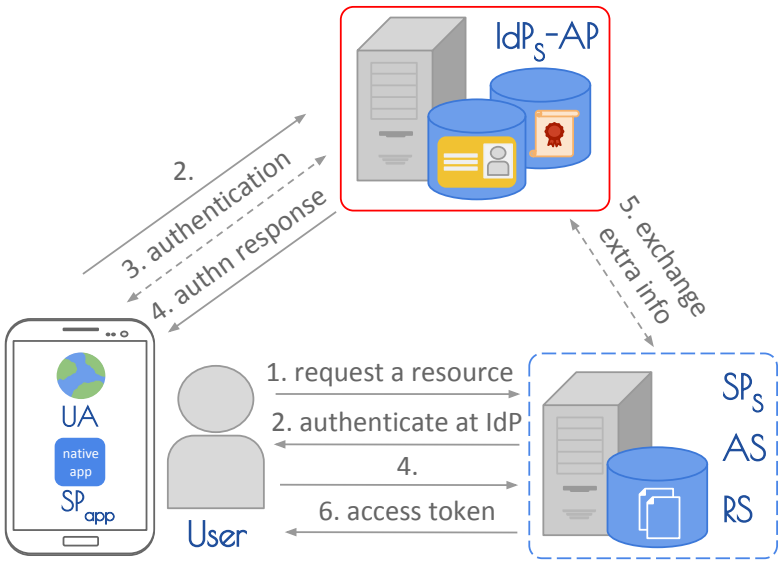


Figure 2.1: Example of IdM Flow.

6. If *User* has the correct attributes, *AS* releases to SP_{app} an access token that can be used to access the protected resource in *RS*.

2.1.1 OTP Generation Approaches

There are many multi-factor techniques on the market. In this work, we focus on a well-accepted solution that combines a PIN code ("something only the user knows") with the generation of an OTP using a software OTP generator ("something only the user possesses"). When an OTP algorithm is used, a different password is generated for each authentication request and is valid only once, providing a fresh authentication property. Thus, compromising an old OTP does not have security consequences in the authentication process.

There exist many algorithms for generating OTPs and we can classify them into three main approaches:

- *Time synchronization:* the OTP is generated starting from a shared secret key (called *seed*) and the current time of the operation. IdP_S

must validate this value: only OTPs that fall into a short temporal range are accepted.

- *Lamport's algorithm* [79]: the first OTP is generated from a seed value and each successor OTP value is based on the value of its predecessor. For example, if s is the seed value and $F(x)$ is a one-way function, we have the following OTPs: $o_1 = s, o_2 = F(o_1), o_3 = F(o_2), \ldots, o_n = F(o_{n-1})$. The last OTP, o_n, is stored on IdP_S. When a *User* wants to login, she sends o_{n-1} to the server, and the server applies the function F and checks that the result corresponds to the stored value. If the two values correspond, IdP_S authenticates *User* and updates the stored value with o_{n-1}. In the next login, *User* will use o_{n-2} and so on. After n logins, *User* has to change the seed value to calculate new OTP values.

- *Challenge/Response:* in the execution of this approach, IdP_S presents a "challenge" (e.g., a random number) and *User* answers with a valid "response", which is an OTP value calculated using a mathematical algorithm starting from the challenge.

In Sect. 7, we will detail and analyze the time synchronization and challenge/response approaches in the context of two real use-case scenarios. We have left aside the Lamport's algorithm as S/KEY (a OTP system based on Lamport's algorithm) is falling into disuse and there are some vulnerabilities (e.g., man in the middle and collisions) that threaten its security.

2.2 IdM Standards

In the following, we will describe some of the most widespread standards on authentication and authorization, namely SAML 2.0 (Sect. 2.2.1), OAuth

2.0 (Sect. 2.2.2) and OpenID Connect (Sect. 2.2.3). We will also present the current support of these standards to the mobile context (Sect. 2.2.4).

2.2.1 SAML 2.0

Security Assertion Markup Language 2.0 (2005 - hereafter SAML) [92] is among the most widespread standards used to exchange authentication and authorization assertions (in XML formats). SAML requires that *User* (called *Principal*) is registered with an *IdP*, which after receiving a request message from a *SP* will respond with an authentication result (called *assertion*). An *assertion* contains data used by *SP* to decide whether to grant or deny for that *Principal* the access to a particular resource. A SAML assertion can contain three types of statement: *authentication statement* that indicates if a user is authenticated; *attribute statement* that describes some of the user attributes; and *authorization decision statement* that specifies whether a user is authorized to do a specific action on a specific resource. SAML defines different protocols, bindings and profiles. A protocol specifies which requests and responses are exchanged. A binding describes how SAML requests and responses are mapped into the communication and transmission protocols (SOAP or HTTP). A profile specifies a use-case scenario choosing a combination of assertions, protocols and bindings. Figure 2.2 illustrates these concepts in a clarifying picture.

Among the profiles defined in the standard, the most used is the *web browser SSO profile* [16]. A SSO protocol enables *Users* to login to an *IdP* only once and gain access to several *SPs* without requiring to authenticate for each one of them. Considering SP-Initiated SSO with Redirect/POST bindings, the flow consists of the following steps (illustrated in Figure 2.3):

S1. *Principal*, through *UA*, asks *SP* to access a resource (identified with an *URI*);

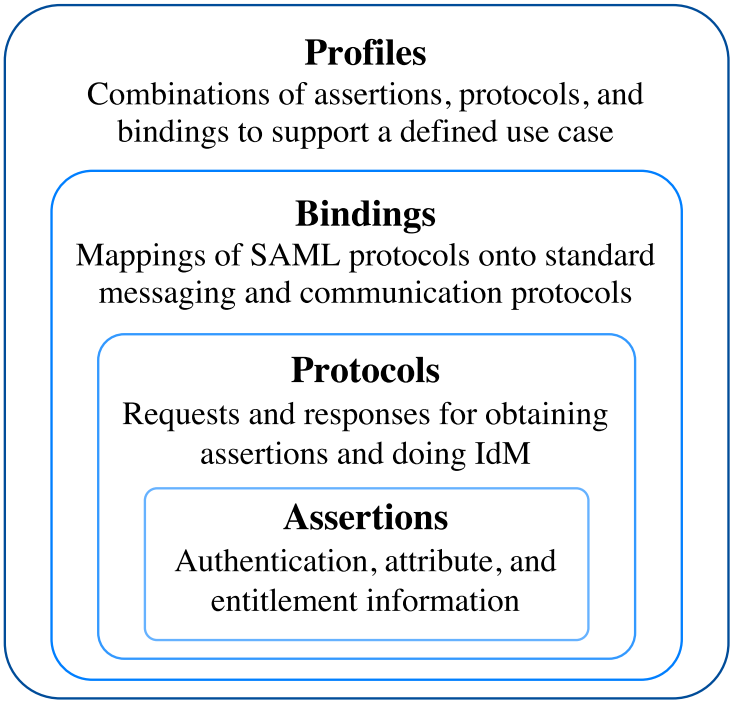


Figure 2.2: SAML Concepts (adapted from [92]).

A1.-A2. In order to grant or deny the access, SP sends (passing through UA) an authentication request $SAMLRequest = (ID, SP, URI)$ to IdP — where ID is a string uniquely identifying the request;

A3.-A4. IdP authenticates $Principal$ (e.g., by providing username and password plus additional authentication factors — notice that the SAML standard is agnostic about the particular authentication process used) and — if the authentication succeeds — sends (passing through UA) the following SAML authentication response $SAMLResponse = (ID, SP, IdP, \{ID, User, IdP, SP\}_{K_{IdP}^{-1}})$ to SP. Where $\{ID, User, IdP, SP\}_{K_{IdP}^{-1}}$ represents the SAML assertion digitally signed with the private key of IdP (K_{IdP}^{-1}).

S2. If the $SAMLResponse$ is valid, SP releases the requested resource to $Principal$.

There are many studies in the literature, such as [30], which focus on the

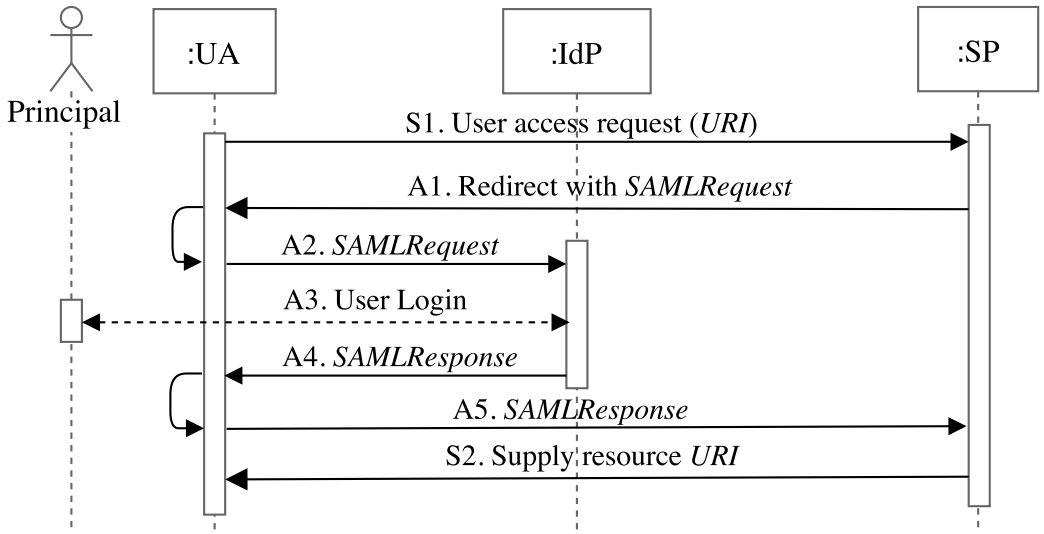


Figure 2.3: SAML Web Browser SSO Profile (adapted from [92]).

analysis of the different SAML SSO implementations and the description of common vulnerabilities and attacks caused by incorrect implementation assumptions. Thanks to the contribution of these security formalizations, SAML is considered a successful standard and is widely used in the corporate environment.

2.2.2 OAuth 2.0

In recent years, in the context of social and mobile applications, we have observed a transition from SAML to *OAuth 2.0* (2012 - hereafter OAuth) [64]. Compared to SAML, OAuth is an authorization standard typically used to manage access delegation.

OAuth introduces different names for the roles that we have defined in Sect. 2.1 (cf. Table 2.2): *User* is called *Resource Owner* (*RO*), the *SP* is called *Client* (*C*), and *IdP* usually corresponds to *AS* — however the authentication process is not specified, thus also external *IdPs* are possible. To illustrate a simple scenario where OAuth is a key enabler, let us consider *RO* which wants to use a *C* app to print some photos (*resources*) stored on

a different photo-sharing site (RS). Using OAuth, RO can grant C access to her private photos, without having to share her credentials by directly authenticating with a server trusted by the photo-sharing service (AS), which issues a token (called *access_token*) to C carrying the requested authorization delegation. The default *access_token* type used in the OAuth standard is a *bearer token*, which in [65] is defined as "a security token with the property that any party in possession of the token (a "bearer") can use the token in any way that any other party in possession of it can. Using a bearer token does not require a bearer to prove possession of cryptographic key material (proof-of-possession)". This means that an entity in possession of an *access_token* can access the corresponding RO resources without any further authentication and authorization processes.

As in different scenarios the entities involved and relative assumptions change, the OAuth standard defines four different flows: *RO Password Credentials*, *Implicit*, *Authorization Code*, and *Client Credentials*. They have the common goal of releasing a token to C encoding the delegation to access resources, but they differ in the way used by C to obtain this token. Here, we give an idea of the messages exchanged in each flow.

Client Credentials Flow. The *Client Credentials* flow is used when the authorization is limited to resources under the control of C and C is acting on its own behalf. This flow includes two steps: in Step 1, C requests an

Table 2.2: Role Names for the Different Standards.

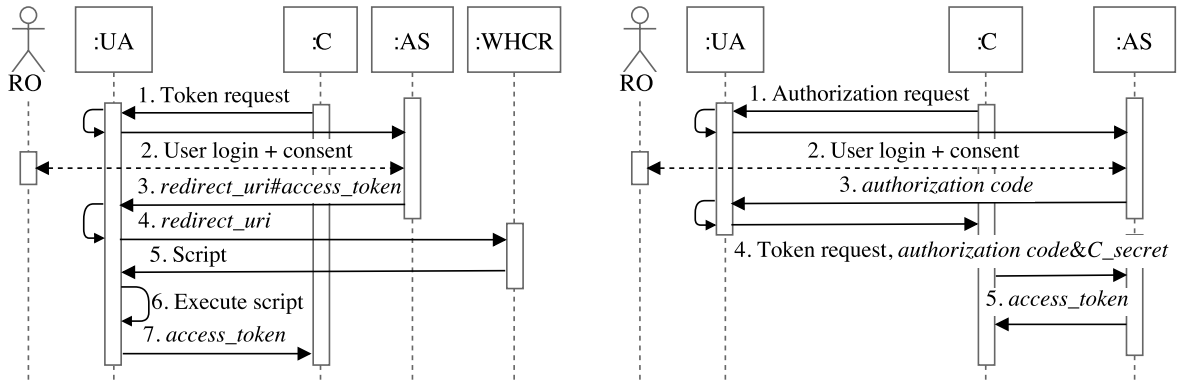
Role of Fig. 2.1	**SAML**	**OAuth**	**OIDC**
User	*Principal*	*RO*	*End-User*
SP_{app}	*SP*	*C*	*RP*
IdP_S	*IdP*	*AS*	*OP*
UA	*UA*	*UA*	*UA*

access_token and, in Step 2, *AS* authenticates *C*, and if valid, issues an *access_token*.

RO Password Credentials Flow. This flow includes the following steps: in Step 1, *RO* provides *C* with her own credentials; then, in Step 2, *C* requests an *access_token* from *AS*'s token endpoint by including the credentials received from *RO*. Finally, in Step 3, *AS* validates the *RO* credentials, and if valid, issues an *access_token*.

Implicit Flow. This flow, illustrated in Figure 2.4(a), involves the following steps: in Step 1, *C* initiates the flow by directing *RO*'s *UA* to the authorization endpoint. *C* also includes a redirection URI (*redirect_uri*) to which *AS* will send *UA* back once access is granted (or denied). In Step 2, *AS* authenticates *RO* (via *UA*) and establishes whether *RO* has granted or denied *C*'s access request. Assuming *RO* has granted the access, *AS* redirects *UA* back to *C* (Step 3). The *redirect_uri* includes the *access_token* in the URI fragment. In Step 4, *UA* follows the redirection instructions by making a request (which does not include the fragment) to the Web-Hosted Client Resource (*WHCR*) that is a web server or a local resource accessible to *UA* and capable of extracting the access token. *UA* retains the fragment information locally. In Step 5, *WHCR* returns a web page (typically an HTML document with an embedded script) capable of accessing the full redirection URI including the fragment retained by *UA*. *UA* executes the script locally (Step 6), which extracts the *access_token*. In Step 7, *UA* passes the *access_token* to *C*.

Authorization Code Flow. This flow, illustrated in Figure 2.4(b), involves the following steps: Steps 1-3 are the same as Steps 1-3 in the Implicit flow (described before). The difference is that the redirection URI now



(a) OAuth Implicit Flow.

(b) OAuth Authorization Code Flow.

Figure 2.4: OAuth Flows (adapted from [64]).

includes an *authorization code* instead of directly an *access_token*. In Step 4, C requests an *access_token* by authenticating with AS using its secret (*C_secret*) and presenting the *authorization code*. AS authenticates C and validates the *authorization code*, and if valid, in Step 5 issues an *access_token*. We observe that the Authorization Code flow is the only OAuth flow that requires the authentication of C. In the last request of the flow, C exchanges an *authorization code* and its *C_secret* to obtain an *access_token* in a direct communication with AS, where *C_secret* is a value obtained in the registration phase and known only by AS and C itself.

Below, we report the appropriate flow choices according to the standard in different scenarios. The Client Credentials flow is used when the authorization scope is limited to the protected resources under the control of C and C is acting on its own behalf (C is also RO). An example of use-case for this flow is server to server communication, as there is no user involvement and the server can be treated as both RO and C. When RO and C are played by two different entities we have to choose the appropriate OAuth flow depending on the client type and profile. RO Password Credentials flow is suitable in cases where RO has a trust relationship with C, such as when C is the device operating system or a highly privileged

application. In the case of a web application, the OAuth protocol suggests the use of the Authorization Code flow, since it is suitable for clients that can keep their credentials confidential when authenticating with *AS*. Moreover, this flow provides some important security benefits when compared with other flows, including the ability to authenticate *C*, and the transmission of the access token directly to *C* without passing it through *UA* and potentially exposing it to others, including *RO*. In the case of *UA*-based applications, the OAuth protocol suggests to use the Implicit flow, that is a simplified Authorization Code flow, optimized for clients implemented in a browser using a scripting language. In the case of mobile applications, the protocol leaves a certain degree of freedom regarding how to implement and apply the standard.

Note that SAML and OAuth solve different problems. SAML aims to exchange assertions about user digital identity, giving *SP* a way to grant or deny access to a resource or authenticate the *User*. OAuth is a standard that manages the delegation of accesses. The basic idea is that *User*s can allow a *SP* to access their resources hosted on another site. Thus, OAuth is more suitable for REST-based systems (e.g., social network APIs), where a *User* wants to allow a *SP* to access an API resource and perform actions on her behalf. The problem is that OAuth is currently used by major *SP*s to support SSO, even though OAuth was not originally designed to manage authentication. The vulnerabilities related to the use of OAuth as an authentication protocol have been analyzed in [39]. This study also points out the main properties that an authentication protocol has to satisfy. *OpenID Connect* solves this particular problem, as it enforces an authentication layer on top of the OAuth authorization standard.

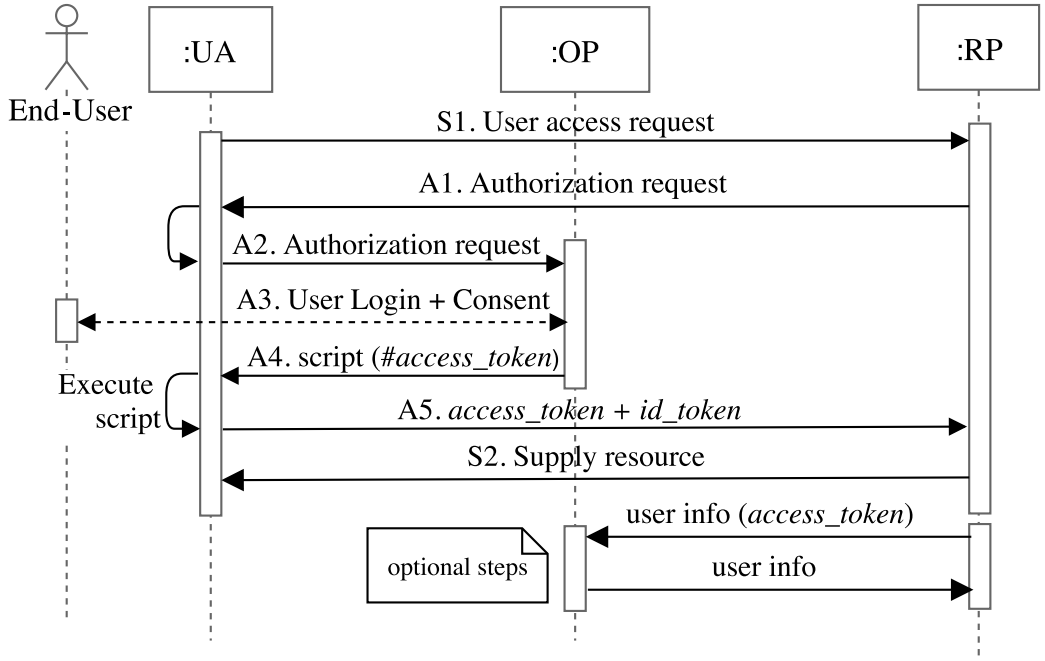


Figure 2.5: OIDC Implicit Flow (adapted from [94]).

2.2.3 OpenID Connect

OpenID Connect 1.0 (2014 - hereafter OIDC) [94], adds to the OAuth standard two features: the *id_token*, a structured JSON token (JWT) that carries information about the authentication process, such as the status of the user's identification; and the *userInfo endpoint*, a REST API for identity attributes, such as email, name and address.

In [105], a comparison between SAML and OIDC is performed. The two standards were designed for the same purpose; they both use signed assertions to exchange claims about the user's session information, but they achieve this using different technical means, the message format being an example: JSON and REST for OIDC versus XML and SOAP for SAML.

OIDC — being based on OAuth — inherits all the OAuth flows. In Figure 2.5, we report the OIDC Implicit flow that is easily comparable with the SAML SP-Initiated SSO with Redirect/POST bindings in Figure 2.3. Once again, OIDC uses different names to identify entities playing a similar

role (cf. Table 2.2). A *User* is called *End-User*, *SP* is called *Relying Party* (*RP*), and *AS* (playing also the role of *IdP* and *AP*) is called *OpenID Provider* (*OP*).

2.2.4 Mobile Context

While SAML does not formally support mobile *SPs* — it was designed in 2005, when mobile apps were not in widespread use — in OAuth, and thus OIDC, we can find a first attempt to do this. The main problem is that the specification leaves many implementation choices to developers (frequent use of the expression "out of scope"). This could lead to the implementation of insecure solutions. An in-depth analysis of OAuth in the mobile environment — underlining possible security problems and vulnerabilities — is available in [39] and [100]. In [39], the two main problems addressed are: the use of OAuth as an authentication standard, and the lack of a mechanism to securely perform redirection in mobile platforms. While [39] warns about the need for clearer OAuth guidelines for mobile app developers, [100] proposes a new framework for securing the OAuth flow in smartphones. This framework requires the use of an embedded browser managed by a trusted app. In this way it does not require the download of an app for every different *IdP* and provides the required separation between *C* and *UA*. However, the level of detail was not sufficient to clarify, for example, how this solution prevents phishing and client impersonation attacks (described in Sect. 6.1.2).

NAPPS. A first attempt to define an authentication and authorization standard for mobile apps has been carried out by the OpenID Foundation with the working group NAPPS (Native Applications) [95]. The idea of this model (represented in Figure 2.6) is to introduce a new entity, called *Token Agent* (*TA*), which manages:

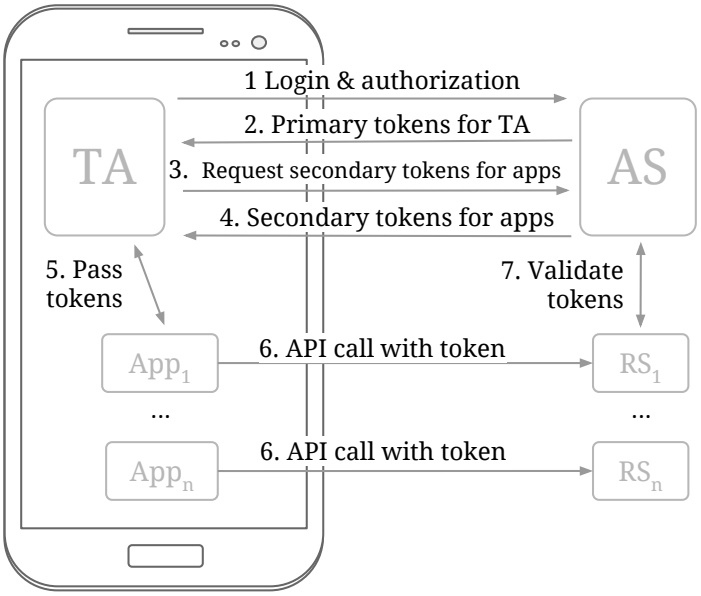


Figure 2.6: NAPPS Flow (adapted from [95]).

Token exchange. Each native application does not directly obtain its own tokens from *AS*, but will receive them from *TA* (*secondary tokens*).

Authentication and authorization process. Users do not need to authenticate in each native application, they do so only for *TA* (providing a SSO experience).

However, the OpenID Foundation has abandoned this specification, and being only a draft version, NAPPS lacks a security analysis and many technical details.

OAuth for native apps. Over the last few months, we have witnessed the release of a new component integrated in the mobile Operating System (OS), having an impact on native SSO solutions. This has brought the different working groups of OAuth and OpenID Connect to modify their preliminary proposals for the native SSO solution.

Both iOS and Android have introduced a new component called *in-app browser tab* which is defined in [93] as "a full page browser with limited nav-

igation capabilities that is displayed inside a host app, but retains the full security properties and authentication state of the system browser". In-app browser tab has platform-specific product names, such as *SFSafariViewController* for iOS and *Chrome Custom Tab* for Android. As pointed out in [84, 83], this new mobile OS component has introduced a new mechanism, which is a middle ground between the two browser solutions currently used (embedded and external browser), to implement browser-based protocols and, consequently, for supporting native SSO. As a result, a new draft for native SSO has been released by the OAuth working group with the title "OAuth 2.0 for Native Apps" [93].

The focus of [93] is to detail the security and usability reasons to prefer a particular implementation of native SSO. Figure 2.7 shows the recommended flow, which corresponds to the OAuth Authorization Code flow. As *UA*, they recommend an external browser, by preferring for usability reasons the in-app browser tabs. In details:

1. C sends an authentication request to *IdP* passing through *UA*.

2. AS receives the authentication request, and processes it, typically by obtaining from *User* her credentials and a permission to allow C to perform the login phase.

3. AS, after having checked the *User*'s credentials and consent, issues a nonce, called *authorization code*.

4. C presents the *authorization code* at the token endpoint of AS.

5. The token endpoint validates the *authorization code* (by checking that the *authorization code* was not expired, was not already used and was issued to the same C that is requesting the token) and issues a $token\_C$.

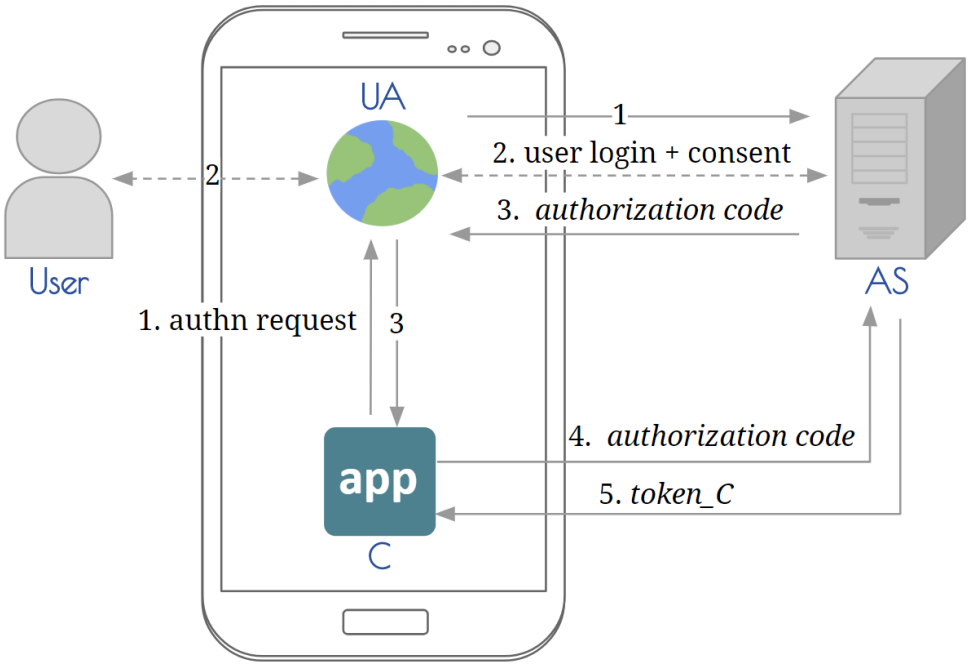


Figure 2.7: OAuth for Native Apps (adapted from [93]).

In [93], they do not specify a unique solution. Actually, for the redirection of the *authorization code* to *C* (Step 3 of Figure 2.7), they propose three different solutions:

1. *App-declared Custom URI scheme.* "Using this inter-app communication, an app can register with the OS a custom URI scheme. When the browser or another app attempts to follow a custom URI scheme, the app that registered it is launched to handle the request."

2. *App-claimed HTTPS URI scheme.* "If supported by the OS[1] an app can claim an HTTPS URL. As before, the URL is opened in the corresponding native app."

3. *Loopback URI redirection.* "It allows apps to create a local HTTP listener on a random port, and receive URI redirects that way."

[1]This solution is currently available only for Android 6 (and above) and iOS 9 (and above).

In this work, we do not analyze *loopback URI redirection* as this approach is mostly used for desktop apps. In addition, as described in [38] this approach does not authenticate the app as any app can create a connection with the same listener.

About the *App-declared Custom URI scheme*, as we will describe in Sect. 4.1.2, a known limitation is that the registration of a specific URI is not unique. Thus, a malicious app can register to the OS the same URI of another app. In this scenario, after a run of the protocol the malicious app could receive the *authorization code* and exchange it to obtain a *token_C*. To mitigate this, the OAuth working group recommends a mechanism that protects the *authorization code* from being used in case it is in possession of a wrong C app. That is the combination of the flow of Figure 2.7 with the PKCE (Proof Key for Code Exchange) standard [69], in this case: the C app creates and stores a value called *code_verifier*, chooses a transformation method t_m and calculates its transformed version $t_m(code\_verifier)$. In Step 1, the C app sends together with the authentication request the parameters $t_m(code\_verifier)$ and t_m. The AS stores these two parameters and responds with an *authorization code*. In Step 4, together with the *authorization code*, the C app sends the *code_verifier* value. Before answering with a token, AS must transform *code_verifier* with the t_m method and compare it with the $t_m(code\_verifier)$ value. If they are equal, a *token_C* is released. The idea is that a malicious app that intercepts an *authorization code* in Step 3 cannot reuse it to obtain a *token_C*, as it does not know the *code_verifier* value. This solution, even with this mitigation, does not specify a C authentication procedure. So, a *User* can perform a run of this protocol with a malicious app without being aware of it. This problem is solved using *App-claimed HTTPS URI scheme*. In this case, the OS checks, after the installation of the app, that the app has the right to claim a specific URL, in this way checking the app identity. In this case,

the use of PKCE is not necessary.

Finally, we observe that even if the OAuth working group solution is based on the OAuth Authorization Code flow they do not exchange the client secret. Indeed, in the case of native apps, as every information is visible by the *RO* (the owner of the smartphone where the native apps are installed), the confidentiality of the client secret is not guaranteed. These types of C, for which the client secret is not confidential, are called *public*. If every information passes through the *RO*'s smartphone, the exchange of the client secret for an access token just makes the flow more complex without adding more security.

An implementation of the flow of Figure 2.7 with *App-claimed HTTPS URI scheme* or *App-declared Custom URI scheme* with PKCE is available at `https://appauth.io/`, where the open source libraries *AppAuth* can be downloaded for Android and iOS.

Chapter 3

Identification of Technical Guidelines and Laws

In this chapter, we present guidelines, laws and regulations that are particularly relevant to digital identity and privacy in the European and Italian contexts. In Sect. 3.1, we define the basic concepts of IdM solutions by referring to the IdM technical standards and recommendations. Then, we identify the IdM and privacy laws and regulations in the European and Italian contexts (Sect. 3.2). Finally, for the Italian scenario, we report a decisional diagram that could be used to identify the legal obligations a specific IdM solution has to be compliant with (Sect. 3.2.3).

3.1 ISO/IEC Standards and Technical Guidelines

Many concepts related to IdM solutions are well-defined in the industrial and commercial standards, e.g., the standards published by ISO/IEC (International Organization for Standardization/International Electrotechnical Commission) in the areas of electronic and electrical technologies. In the IdM context, these ISO/IEC standards must be taken into consideration:

- *ISO/IEC 24760 (A framework for identity management)* [71] gives the

definitions of basic IdM concepts, such as:

  - *Entity* is an "item inside or outside an information and communication technology system, such as a person, an organization, a device, a subsystem, or a group of such items that has recognizably distinct existence", i.e. a human subscriber to a telecom service, a government agency, a SIM card, a passport and a website.
  - *Attribute* is a "characteristic or property of an entity that can be used to describe its state, appearance, or other aspects", i.e. address information, telephone numbers, a privilege, a MAC address and a domain name.
  - *Digital Identity* is "a set of attributes related to an entity (person or organization) in a specific domain".
  - *Identity Provider* is "an entity that makes available identity information".
  - *Federated Identity* is "an identity for use in multiple domains, which together form an identity federation".
  - *Identity Federation* is "an agreement between two or more domains specifying how identity information will be exchanged and managed for cross-domain identification purposes".

- *ISO/IEC DIS 29115 (Information technology - Security techniques - Entity authentication assurance framework)* [72]. This document defines the Level of Assurance of an IdM solution:

  > *Level of Assurance (LoA)* describes the degree of confidence in the processes leading up to and including the authentication process itself, thus providing assurance that the entity

claiming a particular identity is in fact the entity to which that identity was assigned.

The LoA definition involves the following processes: *(i)* enrolment (e.g., identity proofing and registration); *(ii)* credential management (e.g., credential issuance); and *(iii)* authentication. Four different LoA are specified:

LoA1 *Low* - "Little or no confidence in the asserted identity. There is no specific requirement for the authentication mechanism used. This level does not require use of cryptographic methods."

LoA2 *Medium* - "Some confidence in the asserted identity. Single-factor authentication is acceptable."

LoA3 *High* - "High confidence in the asserted identity. This LoA shall employ multi-factor authentication. Identity proofing procedures shall be dependent upon verification of identity information. Any secret information exchanged in authentication protocols shall be cryptographically protected. There are no requirements concerning the generation or storage of credentials; they may be stored or generated in general purpose computers or special purpose hardware."

LoA4 *Very High* - "Very high confidence in the asserted identity. LoA4 is similar to LoA3, but it adds the requirements of in-person identity proofing for human entities and the use of tamper-resistant hardware devices for the storage of all secret or private cryptographic keys.".

Choosing which LoA is appropriate in a specific scenario depends on many factors, usually the final decision is taken based on the evaluated risk. In [102], a technique for doing this risk assessment is suggested. We report in Table 3.1 the proposed (possible) mapping between the potential impact and the appropriate LoA. For example, if the impact that an authentication error causes a financial loss is low then a LoA1 is adequate.

Together with ISO/IEC standards, it is important to follow the security and privacy guidelines on IdM released by organizations such as ENISA (European Union Agency for Network and Information Security) and NIST (National Institute of Standards and technology). Important documents are [90, 91, 42].

3.1.1 Privacy-by-Design

Goal of privacy-by-design is to identify and mitigate privacy risks. Note that, the keyword "design" does not mean that privacy must be considered only during a design phase, indeed a complete privacy-by-design process takes into consideration all the life-cycle of a product (from the design to the final deployment). Therefore, "design" alludes to the need to carry out proactive rather than reactive compliance with privacy.

Many privacy regulations refer to the principles that were identified in 1980 [20] (and updated in 2013 [21]) by OECD, such as:

Purpose Specification: "before, and in any case not later than at the time data collection it should be possible to identify the purposes for which these data are to be used, and that later changes of purposes should likewise be specified." This means that personal data should not be used for different purposes for which they were collected, except with

Table 3.1: Maximum Potential Impacts for each LoA.

Potential impact for authn errors	**LoA1**	**LoA2**	**LoA3**	**LoA4**
Inconvenience or reputation	Low	Mod	Mod	High
Financial loss or organization liability	Low	Mod	Mod	High
Harm to organization programs or interests	None	Low	Mod	High
Unauthorized release of sensitive info	None	Low	Mod	High
Personal safety	None	None	Low	Mod/High
Civil or criminal violations	None	Low	Mod	High

user consent or different laws.

Individual Participation: "the right of individuals to access and challenge personal data". This means that individuals should have an active role in the management of their own personal data and should be made aware of how these data are disclosed.

Security Safeguards: "Security and privacy issues are not identical. However, limitations on data use and disclosure should be reinforced by security safeguards. Such safeguards include physical measures (locked doors and identification cards, for instance), organisational measures (such as authority levels with regard to access to data) and, particularly in computer systems, informational measures (such as enciphering and threat monitoring of unusual activities and responses to them). It should be emphasized that the category of organisational measures includes obligations for data processing personnel to maintain confidentiality." This means that integrity and confidentiality should be implemented to protect personal data.

The individual participation and self-determination principles are well-embedded in the following definition of privacy [47]: "privacy is the right of an entity — in this context usually a natural person — to decide for itself when and on what terms its attributes should be revealed. Privacy can alternatively be described as the freedom of a natural person to sustain a "personal space", free from interference by other entities. In an IdM context, privacy is mostly used as a synonym of "informational privacy", i.e. the interest of a natural person to control, or at least significantly influence the handling of data about themselves, also taking into account the nature of the applicable attributes and the entity in charge of data management."

An important element in the implementation of those principles into

systems are the so-called "Privacy Enhancing Technologies" (PET), which in [35] are defined as "a system of ICT that measures the protection of informational privacy by eliminating or minimising personal data thereby preventing unnecessary or unwanted processing of personal data, without the loss of the functionality of the information system". Examples of PET are encryption, protocols for anonymous communications, and attribute based credentials.

A very useful report on privacy that bridges the gap between legal frameworks and available technological implementations was released by ENISA in 2014 [42]. In [42] a thorough inventory of existing approaches and strategies on privacy are described, and in addition, a list of limitations of a privacy-by-design approach is discussed.

3.2 From Technical to Legal Aspects: Europe and Italy

A fundamental step of our methodology is the identification of legal obligations and requirements given a specific application context. As each nation has its different legal obligations, from a given application context it is important to analyze laws and regulation of the country where the solution will be deployed. Indeed, even if the general principles are similar, a solution targeted for US or Asia has to follow different laws compared to a solution for Europe. For privacy principles, a thorough comparison between the EU legal system (discussed in Sect. 3.2.1) and US regulations (e.g., FIPs, Private Act and HIPAA) is provided in [62]. The main difference is that the US Constitution contains no express right to privacy (it is not recognized as a fundamental right).

We will focus on the European directives and regulations on IdM and privacy. Compared to a regulation, which becomes immediately enforce-

able as law in all Member States (MSs), a directive requires the different MSs to have an active role in the achievement of the results (transposition in the national laws). Thus, inside Europe we can also have different legal obligations. We have performed a detailed analysis on Italian laws, extracting the more relevant legal aspects in the context of IdM and privacy to provide a decisional diagram that could be used to evaluate which legal obligations a specific IdM solution has to follow. As we will highlight later, many European and Italian rules on IdM and privacy adapt concepts described in Sect. 3.1. Indeed, to facilitate a correct implementation, legal documents recall — if possible — technical aspects from industrial and commercial standards. In this section, we will analyze the current laws on digital identity and privacy in both the European and Italian framework. First, we give an overview of the EU digital identity and privacy strategies and regulations, and then, present the Italian digital identity and privacy framework.

3.2.1 Digital Identity and Privacy in the EU Law

The theme of digital identity is extremely sensitive in the EU. There are many initiatives and guidelines promoted by EU concerning digital identity, and in particular a key role is played by the European Commission. In 2010, the European Commission launched the Digital Agenda for Europe (DAE),[1] which is one of the initiatives of Europe 2020 Strategy and has the goal of improving the citizens' quality of life through digital technologies by simplifying the access to services of the public administration. One of the key aspects of the DAE is the attempt to create an EU Digital Single Market, which also appears as one of the action points of the

[1] http://ec.europa.eu/digital-agenda/.

Single Market Act[2] introduced by the European Commission in 2011. The idea is to create an area "where individuals and businesses can seamlessly access and exercise online activities under conditions of fair competition, and a high level of consumer and personal data protection, irrespective of their nationality or place of residence"[45]. As a consequence of the Digital Single Market, the DAE decided to expand the existing Directive on digital signatures (Directive 1999/93/EC [49]) introducing a new Regulation (910/2014 [50]), also called eIDAS (electronic IDentification and Authentication Services), which is characterized by an EU cross-border framework. The main reason for the European Commission to introduce a new legislation, specifically a regulation, instead of another directive, is the need to create an EU digital single market, which would thereby establish a legal and technical interoperability among the MSs.

We would like to underline that the digital identity topic is only mentioned in the new Regulation (Chapter II - Electronic Identification). The Directive 1999/93/EC is here cited only because of the relationship between the two legislations; the actual main theme of the Directive is instead the e-Signature.

> *Article 1 (Scope) of Directive 1999/93/EC*: "The purpose of this Directive is to facilitate the use of electronic signatures and to contribute to their legal recognition. It establishes a legal framework for electronic signatures and certain certification-services in order to ensure the proper functioning of the internal market."

In the eIDAS Regulation, many concepts related to the eSignature are reaffirmed, such as the legal effects of the eSignature: a qualified eSignature

[2]More information at `http://ec.europa.eu/digital-agenda/en/digital-single-market` and `http://www.digitalplan.gov.gr/resource-api/dipla/contentObject/Single-Market-ACT-20120206_new_growth_en/content`.

shall be equivalent to an handwritten signature (art. 5, co.1, lett. a, Directive 1999/93/CE and art. 25, co. 2, eIDAS Regulation). We can better understand the differences in the purposes of the two legislations comparing their Article 1.

> *Article 1 (Subject matter) of eIDAS*: "With a view to ensuring the proper functioning of the internal market while aiming at an adequate level of security of electronic identification means and trust services this Regulation:
>
> (a) lays down the conditions under which Member States recognize electronic identification means of natural and legal persons falling under a notified electronic identification scheme of another Member State;
>
> (b) lays down rules for trust services, in particular for electronic transactions; and
>
> (c) establishes a legal framework for electronic signatures, electronic seals, electronic time stamps, electronic documents, electronic registered delivery services and certificate services for website authentication."

Thus, the eIDAS Regulation deals with the eSignature and introduces new rules for digital identity and trust services. We report below the definitions of Article 3 of eIDAS related to digital identity:

- "*electronic identification* means the process of using person identification data in electronic form uniquely representing either a natural or legal person, or a natural person representing a legal person;"
- "*person identification data* means a set of data enabling the identity of a natural or legal person, or a natural person representing a legal person to be established;"

- "*authentication* means an electronic process that enables the electronic identification of a natural or legal person, or the origin and integrity of data in electronic form to be confirmed;"

Comparing eIDAS definitions with the ISO/IEC definitions of Sect. 3.1, we have the following matches: the ISO attributes correspond to the person identification data; and an ISO entity corresponds to a natural or legal person.

To complete the overview of eIDAS definitions related to digital identity, we report below *Article 8 (Assurance levels of electronic identification schemes)* that defines three Levels of Assurances (LoA):

LoA 1 "*Assurance level low* shall refer to an electronic identification means in the context of an electronic identification scheme, which provides a limited degree of confidence in the claimed or asserted identity of a person, and is characterised with reference to technical specifications, standards and procedures related thereto, including technical controls, the purpose of which is to decrease the risk of misuse or alteration of the identity;"

LoA 2 "*Assurance level substantial* shall refer to an electronic identification means in the context of an electronic identification scheme, which provides a substantial degree of confidence in the claimed or asserted identity of a person, and is characterised with reference to technical specifications, standards and procedures related thereto, including technical controls, the purpose of which is to decrease substantially the risk of misuse or alteration of the identity;"

LoA 3 "*Assurance level high* shall refer to an electronic identification means in the context of an electronic identification scheme, which provides a higher degree of confidence in the claimed or asserted identity of a person than electronic identification means with the assurance level

substantial, and is characterised with reference to technical specifications, standards and procedures."

Note that LoA1 (Low) of the ISO/IEC 29115 [72] is not considered in eIDAS. Thus, to do a match between these concepts we have to increase of one the level of ISO/IEC: LoA2, LoA3, LoA4 of ISO/IEC correspond to LoA 1, LoA 2 and LoA 3 of eIDAS, respectively.

Another important aspect, introduced in Article 9, is the concept of *notification*. A Member State (MS) can notify its eID scheme to the European Commission if a detailed list of conditions (see Article 7) is satisfied. When an eID scheme is correctly notified and has the correct assurance level, i.e. it has a LoA equal to or higher than the LoA required to access the service, shall be recognized from all the EU MSs (see Article 6). Note that, the eIDAS Regulation does not intervene in the definition of the different national eID scheme, but it provides a legal framework for the mutual recognition of the digital identity among the MSs, in a way that citizens of a MS can use their own national eID to access services provided by another MS in a secure way. Italy has recently notified its national eID scheme (SPID) to the EU, second country after Germany.

Privacy

As well presented in [61], the first definition of privacy dates back to 1890 due to the work of Warren and Brandeis in the "The Right to Privacy" [109], where privacy is defined as the individual right to be let alone. In the last centuries, the definition of the right to privacy has continually changed, principally due to the technology development. As written in [98]: "the digital revolution involves even the change of the notion itself and of the content of the right to privacy: no more right to be let alone, but the right to maintain control on our data".

In Europe, the privacy principles are defined in different acts, including:

- Directive 95/46/EC of the European Parliament and the Council of 24 October 1995 on the Protection of individuals with regard to the processing of personal data and on the free movement of such data (called Data Protection Directive - DPD);

- Directive 97/66/EC of the European Parliament and of the Council of 15 December 1997 concerning the processing of personal data and the protection of privacy in the telecommunications sector;

- Directive 2002/58/EC of the European Parliament and the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications), abrogating directive of 1997;

- Regulation (EU) 2016/679 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (called General Data Protection Regulation - GDPR).

These acts include the definitions of general principles that establish how data can be processed, such as purpose specification, consent and minimal disclosure, and specify regulations on personal and sensitive data. A thorough description of these principles can be found in [61].

eIDAS and Privacy. Regarding privacy, in Article 5, coma 1, eIDAS states that the processing of personal data shall be carried out in accordance with Directive 95/46/EC of the European Parliament and the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. Thus, all the principles related to the Data Protection laws, such as minimal

disclosure, purpose specification and consent, must be directly applied to the eID scheme solutions.

3.2.2 Digital Identity and Privacy in the Italian Law

The entity responsible for carrying on the principles exposed by the DAE in Italy is AgID (Agenzia per l'Italia Digitale).[3] AgID is part of the Presidency of the Council of Ministers and has the role of implementing the Italian Digital Agenda, which defines the set of rules for the development of innovation and digital economy in Italy. At the end of 2014, AgID has released a decree on digital identity management, called SPID (Sistema Pubblico per la gestione dell'Identità Digitale [9]) that is actively used from 2016. The Article 17-ter of the "Decreto del Fare" modifies the coma 2 of the Article 64 of the CAD (Codice per l'Amministrazione Digitale [36]), concerning the modalities of access to the online services released by the Public Administration (PA), by introducing SPID. Italy already disposes of different tools for the digital identification: CIE (Carta d'Identità Elettronica), CNS (Carta Nazionale dei Servizi), TS-CNS (Tessera Sanitaria-CNS), DDU (Documento Digitale Unificato) and now SPID. We will now try to clarify the relationships among them:

- In 2000, CIE was introduced as a smart-card which could serve both as a personal identification card, replacing the old paper-based ID card, and as a tool for the remote access to digital services of the PA. However, as severe obstacles emerged regarding the implementation and actual use of the CIE, in order to guarantee access to the public digital services, the CNS was thereafter introduced;[4]
- CNS is a smart-card which serves as an access tool to the PA online

[3] `http://www.agid.gov.it/`.

[4] A new version of CIE (called CIE 3.0) will be available for all the Italian citizen before the end of the 2018. More details on CIE 3.0 and its application could be found in Sect. 7.3.

services but not as an identity document (it does not include a photo of the owner and does not support particular characteristics for the graphic support). At the time of the deployment of the CNS, the Ministry of Economy and Finance introduced the TS, a health insurance card that replaced the old Italian fiscal code card; the TS-CNS was none other than the result of a collaboration between the PA and the Ministry of Economy and Finance;

- The DDU was conceived in order to simplify and harmonize the current system and replacing all the previous cards; in this way, a unique card would concentrate the same functionality and provide a compatible layout of both the CIE and TS-CNS cards.

SPID subsumes these tools. To obtain a SPID digital identity a citizen has to proof her identity through one of the following five ways (described in Art. 7 of DPCM SPID): *(i)* with an in-person identification *(ii)* with a remote identification using the DDU, CIE or CNS having a valid certificate, or *(iii)* with a remote identification through another SPID identity possessing a security level equal or higher than that being applied for.

The purpose of the SPID system is explained in the following article:

> *Article 2, comma 2*: "SPID enables users to make use of digital identity managers and qualified attribute managers to enable suppliers of services the immediate verification of their own identity and of any qualified attributes that concern them."

Thus, the SPID system aims to make the access and use of public and private digital services easier and safer.

SPID and eIDAS. The SPID system is compliant with the eIDAS Regulation, so it will most likely be accepted by the other EU MSs. Likewise,

the SPID defines three security levels (SPID 1, SPID 2 and SPID 3) referring to the ISO/IEC 29115 that roughly match the eIDAS levels (low, substantial and high).

Privacy

In Italy, the legal aspects related to privacy are covered by the "Personal Data Protection Code" (legislative Decree no. 196/2003) [57] and its Annex B (Technical Specifications Concerning Minimum Security Measures). We report here, the main sections of the Personal Data Protection Code related to authentication:

- *Section 31 (Security Requirements)*: "1. Personal data undergoing processing shall be kept and controlled, also in consideration of technological innovations, of their nature and the specific features of the processing, in such a way as to minimise, by means of suitable preventative security measures, the risk of their destruction or loss, whether by accident or not, of unauthorized access to the data or of processing operations that are either unlawful or inconsistent with the purposes for which the data have been collected."

- *Section 33 (Minimum Security Measures)*: "1. Within the framework of the more general security requirements referred to in Section 31, or else provided for by specific regulations, data controllers shall be required in any case to adopt the minimum security measures pursuant either to this Chapter or to Section 58(3) in order to ensure a minimum level of personal data protection."

- *Section 34 (Processing by Electronic Means)*: "1. Processing personal data by electronic means shall only be allowed if the minimum security measures referred to below are adopted in accordance with the

arrangements laid down in the technical specifications as per Annex B:

a) computerised authentication;

b) implementation of authentication credentials management procedures;

c) use of an authorisation system;

d) regular update of the specifications concerning scope of the processing operations that may be performed by the individual entities in charge of managing and/or maintenancing electronic means;

e) protection of electronic means and data against unlawful data processing operations, unauthorised access and specific software;

f) implementation of procedures for safekeeping backup copies and restoring data and system availability;

g) [Repealed by Section 45(1)c. of decree 5/2012 as subsequently converted, with amendments, into Act no. 35 dated 4 April 2012.]

h) implementation of encryption techniques or identification codes for specific processing operations performed by health care bodies in respect of data disclosing health and sex life. (...)"

While in Annex B, the following security measures are specified:

- *Point 2.* "Authentication credentials shall consist in an ID code for the person in charge of the processing as associated with a secret password that shall only be known to the latter person; alternatively, they shall consist in an authentication device that shall be used and held exclusively by the person in charge of the processing and may be associated with either an ID code or a password, or else in a biometric feature that relates to the person in charge of the processing and may be associated with either an ID code or a password."

- *Point 5.* "Where provided for by the relevant authentication system, a password shall consist of at least eight characters; if this is not allowed by the electronic equipment, a password shall consist of the maximum permitted number of characters. It shall not contain any item that can be easily related to the person in charge of the processing and shall be modified by the latter when it is first used as well as at least every six months thereafter. If sensitive or judicial data are processed, the password shall be modified at least every three months."

SPID and Privacy. The SPID system implements different solutions to be compliant with the Data Protection law. In particular, the principles of minimal disclosure, purpose specification and consent are considered. Indeed, only the user data needed for the specific transition are revealed to the service provider; for example, if a service needs to know if a user has more than 18 years, then only that information and no other attributes, such as the name and the address, would be revealed. Clearly, if an attacker obtains a SPID digital identity, then a great amount of user sensitive information would be revealed. SPID classifies attributes into three different types (Art 1, coma 1, letters c, d, and e):

Identifying attributes: name, surname, place and date of birth, sex, or business name, registered offices as well as the tax identification code or the VAT number and the details of the identity document used for identification;

Secondary attribute: mobile telephone number, email address, physical residence and digital address, as well as any other attributes identified by the Agency useful for communication;

Qualified attributes: qualifications, professional certifications and powers of representation and any other types of attributes certified by a

qualified attribute manager.

3.2.3 Decisional Diagram: Italian Case

As many variables are involved, to understand which legal obligations follow is often not simple. We have done this exercise in relation to authentication legal obligations in Italy, the corresponding decisional diagram is shown in Figure 3.1. As each European Member State has its different legal obligations, it is very challenging to have a unique decisional diagram for the European case[5], however equivalent decisional diagrams can be obtained analyzing laws of different countries and using this scheme as a starting point.

As the different Italian IdM laws regulate only the privacy related to personal data, the first distinction of Figure 3.1 is given by the data type processed. Whether the data processed are not personal, there are no legal obligations to follow.

The second distinction regards who process the data: a Public Administration (PA) or a private company. In case of PA, the CAD (Codice dell'Amministrazione Digitale - D.Lgs.n. 82/3005) [36] must be followed. In article 64, CAD specifies that only CNS, CIE or SPID can be used as login mechanism. While CIE and CNS guarantee a LoA 3 (in these smartcards a certificate is stored), the LoA of SPID can be set based on the specific data nature and the features of the processing. In this case, a developer can follow the classification made by AgID in Annex 4 of SPID [24]. For example, SPID 1 is enough if a citizen is accessing civil consultations, while SPID 2 is required to deal with health data. The Italian "Personal Data Protection Code" (legislative Decree no. 196/2003) [57] specifies that

[5]The eIDAS Regulations itself refers to EU directives (e.g., Directive 95/46/EC on the protection of individuals with regard to the processing of personal data) that need a transposition in the different national laws.

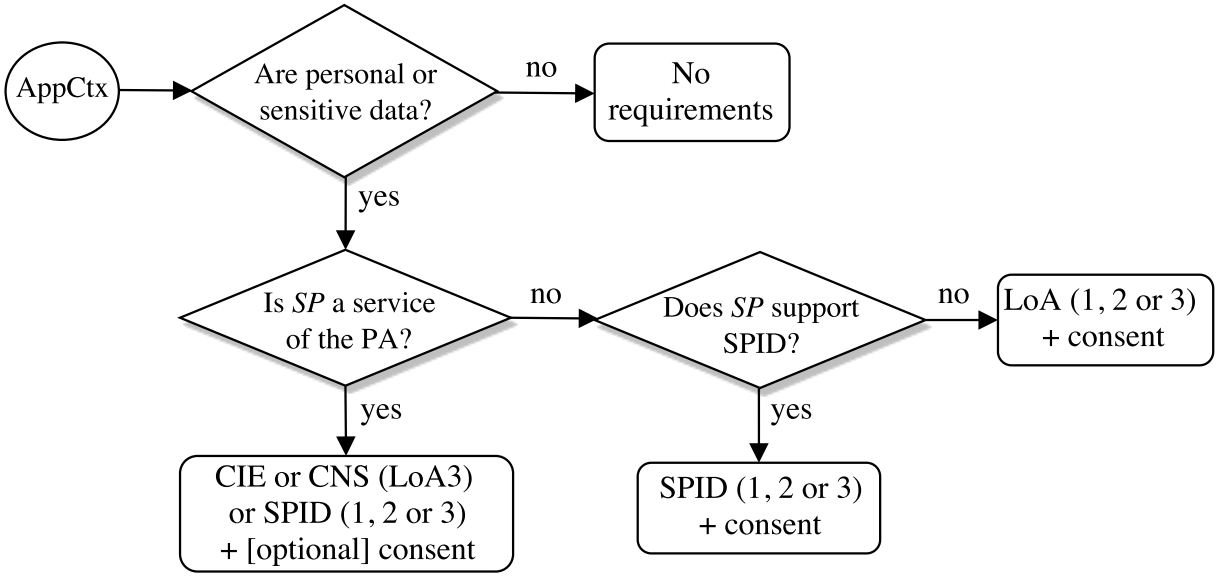


Figure 3.1: Decisional Diagram on Italy Legal Obligations.

for PA the requirement of the user consent depends from the data nature (e.g., for health data is usually required).

In case of private companies dealing with personal data, they can decide to use SPID (after an enrollment process with AgID) or can use other login mechanisms. In both cases the consent of the user is required (cf. [57, Section 23] "Processing of personal data by private entities or profit-seeking public bodies shall only be allowed if the data subject gives his/her express consent"). With SPID, they can consult the Annex 4 of SPID. Using other login mechanisms it is their task to identify which LoA must be implemented based on the data nature and the features of the processing. In Chapter 7, we have identified the required LoA for different real-world scenarios that deal with different data types, from personal data (e.g., name and surname) to sensitive data (e.g., health data).

Chapter 4

A Design Methodology for IdM Solutions

The term *design* is defined in [6] as "the process of deciding how something will be made, including how it will work and what it will look like". There is an emerging common opinion that at the basis of an innovative and successful product there is a high quality design approach, which is able to overcome interdisciplinary constraints. An example of combining design and interdisciplinary fields is the "design thinking" perspective, which — as described in [7] — involves the work of people with different skills with the goal of creating a solution for a real-world problem. A similar design-thinking perspective should be applied to privacy and security. As described in [37], privacy and security should be incorporated in the solution not as an afterthought, but rather, by design.

Security-by-design means that the solution has been designed from the first steps to be secure. In the process of analyzing the security of a proposed design, malicious attackers are taken into account and the global impact is evaluated. The final security-by-design solution is the one that minimizes the impact of the identified security vulnerabilities. Considering security from the early stage of the design is crucial. There are many examples of leaks of personal data caused by an incorrect design. A recent

case is described in [17], the mentioned Italian company was subjected to different attacks that exploit wrong design choices. The article describes a vulnerability that allows an intruder to login as a victim on the online invoice service only knowing the victim's fiscal code (that is a public value). Another example is in [14], where a bug of macOS High Sierra (the latest version of Apple's operating system) lets anyone that has physical access on the target computer to gain root access without a password.

Regarding privacy, we consider the privacy-by-design approach that uses technology as a way to enforce legal obligations. We have identified a set of laws and regulations that must be taken into account dealing with authentication and authorization solutions. As it emerged by the regulations and best practices, there are different privacy principles that should be followed (well-described in Sect. 3.1.1): limitations on data use and disclosure should be reinforced by security safeguards (e.g., integrity and confidentiality), personal data should not be used for different purposes for which they were collected, an individual has the right to determine the disclosure and use of her personal data (self-determination), and so on.

Finally, about usability, we have followed a human-centred design approach that has the goal of making usable solutions, having in mind the user's needs and requirements. We focus on usability as it is strictly linked with security. As we will detail in Sect. 6.3, the highest levels of security can be obtained only with an equally high level of usability. Thus, output of our methodology should be a solution that guarantees a high level of security while maximizing usability. Examples of usable features are the implementation of a Single Sign-On (SSO) experience (*User* has to enter her credentials only once and then can access different services) and the exchange of OTPs in a transparent way (our solutions do not bother *Users* to digit long OTP strings as they are directly communicated by the OTP generator apps).

Even if security-by-design, privacy-by-design and usability are well-accepted concepts their application is not easy and depends on many aspects. In the context of IdM solutions, we have defined a specific design methodology that encompasses these aspects (described in Sect. 4.1).

4.1 Methodology Overview

A main contribution of our research is the definition of a methodology for the design and security assessment of IdM solutions for mobile apps.

Concerning authentication, we consider scenarios where an *identity federation* is established. An identity federation as defined in [71], is "an agreement between two or more domains specifying how identity information will be exchanged and managed for cross-domain identification purposes". There are different levels of identity federation, a widespread federation scenario is the delegation of the authentication process (previously performed by the *SP*) to external *IdPs* (from different security domains). A classic example is the integration of the social login feature (e.g., Sign-in with Google or Facebook) into SP_{app}. In this case, IdP user credentials can be used to gain access to several *SP* apps, and this implies that *Users* do not need to have credentials with a *SP* to access it. We do not consider *SPs* and *IdPs* belonging to the same security domain, as it is relatively easy to design (e.g., exchanging of a cookie).

Regarding security, we focus on the analysis of the design phase, dealing only with implementation aspects that are strictly related to the design (e.g., channel or UA choices). In particular, we will not consider threats related to cryptographic algorithms — which we assume perfect — and implementation aspects (e.g., input validation, SQL injection and DoS attacks). We assume that the common security implementation countermeasures described in standard guidelines (e.g., in the "Digital Identity

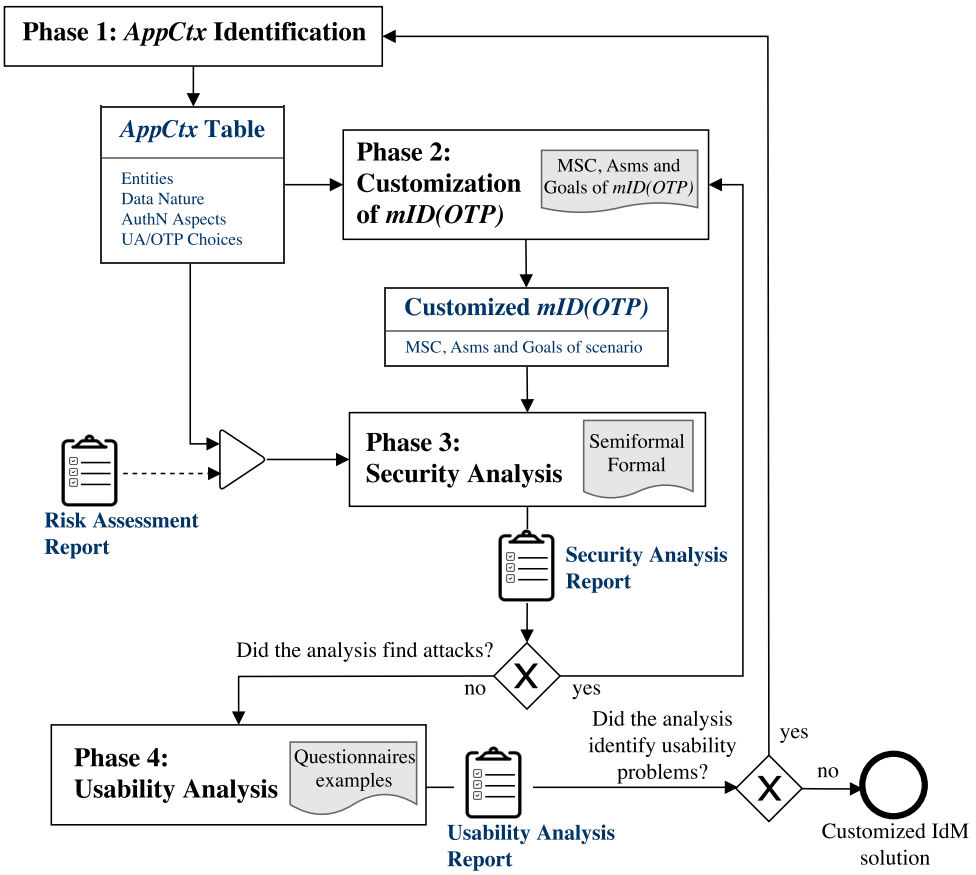


Figure 4.1: Methodology: an overview.

Guideline" of NIST [8] and the ENISA "Smartphone Secure Development Guidelines" [18]) are implemented.

In this context, to obtain a security-by-design, privacy-by-design and usable solution, we have identified four phases: *Application Context (AppCtx) Identification*, *Customization of mID(OTP)*, *Security Analysis*, and *Usability Analysis*. These phases are graphically shown in Figure 4.2 and will be exhaustively described in Chapters 5 and 6. Here we give a global overview considering the simplified representation of Figure 4.1:

Phase 1: *AppCtx* **Identification.** As first phase of our methodology, an IdM designer is required to define the Application Context (*AppCtx*) of the solution she is analyzing. *AppCtx* is derived by an (informal)

description and analysis of the solution, and takes into account the legal aspects described in Chapter 3. In particular, the IdM designer is required to specify the entities involved in the scenario and their mapping among a set of IdM roles we provide, and select a set of options, such as the nature of the processed data (anonymous, personal or sensitive) and the authentication aspects. In Sect. 5.1, we will detail these options, and we will give an example for the Facebook scenario (Sect. 5.1.1). The output of Phase 1 is a table that contains all the information related to *AppCtx*.

Phase 2: Customization of *mID(OTP)*. Designing a security protocol from scratch is not a simple task possibly leading to serious mistakes (we argument this statement in Sect. 4.1.1). We have thus extracted a reference model (flow description and identification of security assumptions and goals) for Android[1] native IdM solutions from the rational reconstruction of Facebook solution (described in Sect. 4.2) and the analysis of OAuth for native app (described in Sect. 2.2.4). We called it *mID(OTP)* to highlight the dual goal that our model pursues: *(i)* "*mID*" represents the management of identities for native mobile apps, providing a Single Sign-On (SSO) experience, and *(ii)* "(*OTP*)" represents the optional establishment of a Multi-Factor Authentication (MFA) which is parametric on the OTP generation approach. In our analysis, we have detailed the OTP generation for TOTP and Challenge-Response (CR) approaches (described in Sect. 2.1.1).

The second phase of our methodology takes as input the *AppCtx* table, and, based on the *AppCtx* values, *mID(OTP)* is automatic customized. If the application scenario requires the interaction with extra

[1]We focus on Android, even if the same concepts can be applied to other operating system such as iOS (cf. Sect. 4.1.2).

entities or requires the integration with OTP generation approaches different from TOTP or CR, then some extra (manual) design is required. The output of Phase 2 is a customized *mID(OTP)* for the scenario, i.e. identification of the Message Sequence Chart (MSC), security assumptions (Asms) and goals specific for the application scenario. We will describe *mID(OTP)* and detail the customization phase in Sect. 5.2.

Phase 3: Security Analysis. Based on the data nature specified as input in the *AppCtx* table and (optionally) on the results of a risk assessment, in this phase an adequate security analysis level is chosen by the IdM designer. Our methodology supports two levels: semi-formal analysis and formal analysis. Chapter 6 presents the details of these two security analyses. Output of this phase is a security analysis report. At this point, the IdM designer has to evaluate the report: did the analysis find attacks? If the answer is no, it means that the design has an adequate degree of security and so the designer can continue with the last phase. Otherwise, Phase 2 must be performed again.

Phase 4: Usability Analysis. The final phase consists of a usability analysis that evaluates the usability aspects of the design and optionally evaluates the user satisfaction using ad-hoc questionnaires (details in Sect. 6.3). Our methodology provides a template example for a usability questionnaire. Output of this phase is a usability analysis report. At this point, the IdM designer has to evaluate the report: did the analysis identify usability problems? If the answer is yes, the IdM designer has to come back to Phase 1 and re-discuss the selected usability aspects. Otherwise, the final customized IdM solution is obtained.

In Chapter 7, to validate our methodology we will apply it to four

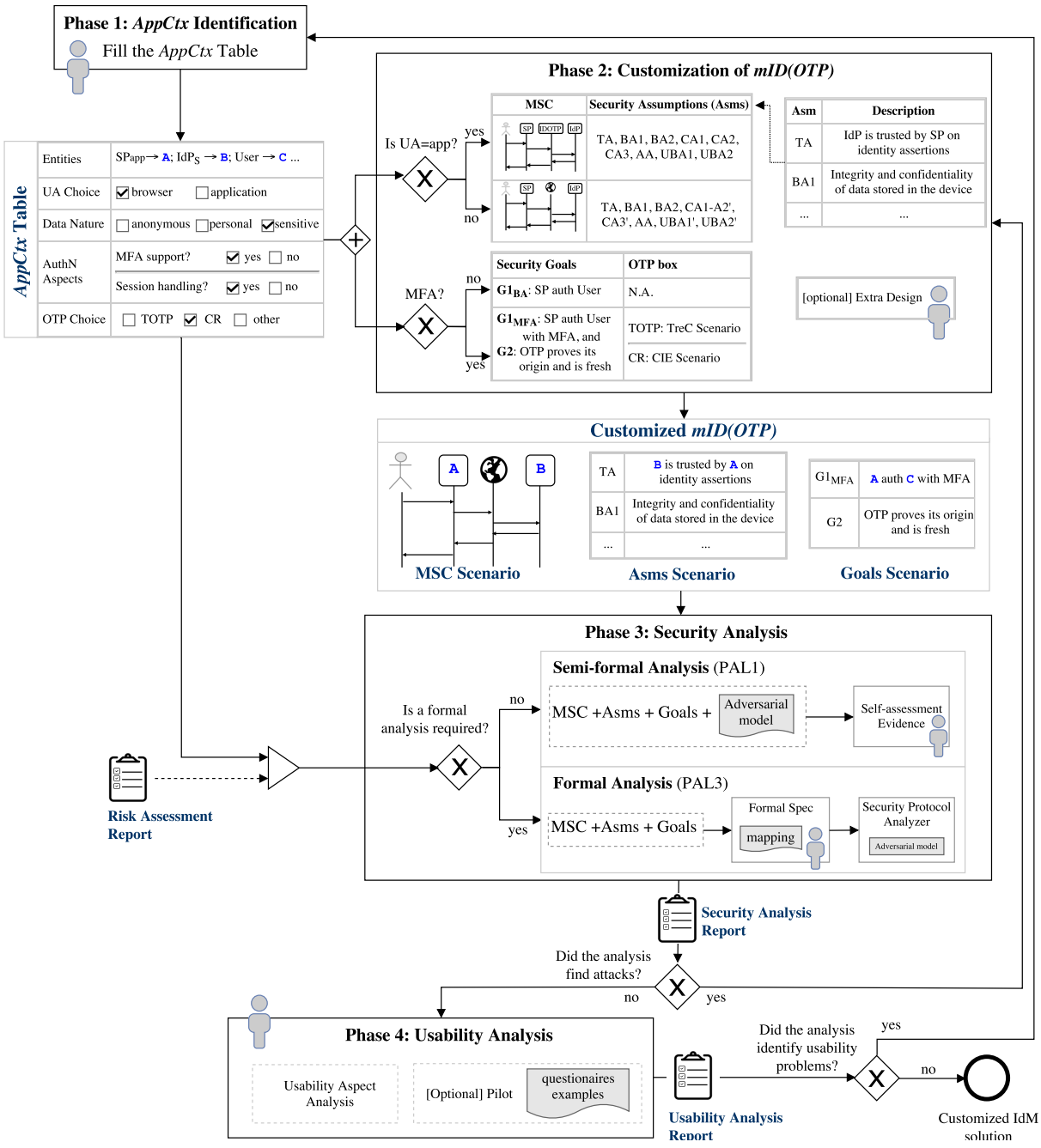


Figure 4.2: Methodology: details.

different real-world scenarios in the context of projects in which we were involved, which consider different authentication and usability aspects:

- *TreC project*: a MFA solution with a SSO experience for mobile e-

Health applications.

- *Smart Community project*: a secure delegated access solution in the context of smart-cities.
- *DigiMat-Lab (Istituto Poligrafico e Zecca dello Stato)*: a mobile MFA solution that uses as second factor the Italian electronic identity card.
- *FIDES EIT activity*: a solution that combines federation and cross-border aspects in the context of the European single digital market.

Before presenting the detailed description of each phases of the methodology, we consider worthwhile to discuss some aspects that are essential to explain our approach. First, Sect. 4.1.1 will discuss the need for a reference model that can be used as a starting point for the design of IdM solutions. Second, in Sect. 4.1.2, we will describe the main Android implementation features that are related to the authentication and authorization aspects and are used by *mID(OTP)*. Then, to complete the overview, in Sect. 4.2, we will present our rational reconstruction of Facebook social login for Android native apps upon which we have based our reference model.

4.1.1 Need for a Reference Model

Having a new IdM solution to model, a first option could be to design the underlying security protocol from scratch. However, examples of even simple security protocols such as Needham-Schroeder [88] and Otway-Rees [96] have been found to be unsound after many years from their adoptions. More recently, the handshake of the WPA2 protocol for Wi-Fi connections was found vulnerable after 15 years from its design [107]. These examples underline how the design of a protocol even of few steps — five in the case of Needham-Schroeder — is not a simple task, as its security depends on several trust and communication assumptions and, in most cases, results in

a solution with hidden vulnerabilities. A good practice is to refer to standards already analyzed by the security community. However, as highlighted in Sect. 2.2, IdM standards for mobile applications are very recent (e.g., the first version of "OAuth for native apps" [93] was published in 2016) and available only in draft versions. In addition, for OAuth, the security community recognized that a "*secure usage of OAuth for mobile applications could be mysterious since the protocol is primarily designed in the mindset of traditional web technology rather than mobile platforms*" [39]. To alleviate the problem, major identity providers (such as Google or Facebook) provide mobile app developers with Software Development Kits (SDKs) to integrate their OAuth-based services together with documents explaining security best-practices on how to implement them. Unfortunately, each provider develops both the SDK and the best practices referring to their implicit security requirements and business logic, making these of little or no use in other contexts. Thus, in our study, we have decided to perform an in-depth analysis on the design and security of state-of-the-art solutions. Starting from the FB solution (presented in Sect. 4.2.2) and the analysis of OAuth for native apps (described in Sect. 2.2.4), we propose a reference model for IdM for native app solutions that can be used as a starting point for the design of an IdM protocol. This model, which is described in detail — with the corresponding MSC, set of assumptions and goals — in Sect. 5.2, is independent from a specific *IdP* and will be used as a blueprint for the design of other IdM solutions. It includes an (optional) MFA based on the generation of OTPs. In Sect. 5.2.4, we compare our reference model with the Facebook solution and the model proposed by the OAuth working group.

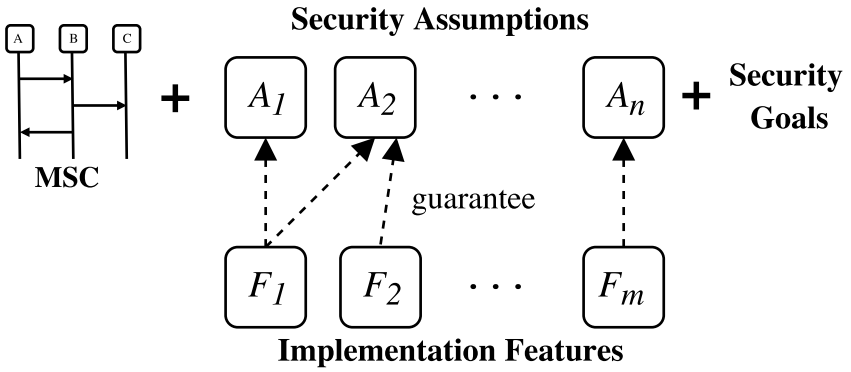


Figure 4.3: Relationship between Implementation and Design.

4.1.2 Implementation Features

The methodology we present deals only with implementation aspects that are strictly related to the design. As shown in Figure 4.3, we have selected as security assumptions $(A_1, A_2, \ldots, A_n)$ a set of implementation features $(F_1, F_2, \ldots, F_m)$ that guarantee these assumptions. Thus, as a consequence, we can infer that — at a design level — the selected implementation features are enough to guarantee the degree of security established during the security analysis.

In this section, we present an overview on Android implementation features that we have taken into consideration in the design process divided in: *(i)* inter-app communication mechanisms detailing URI-scheme and redirections, *(ii)* modalities to check the identity of federated applications, and *(iii)* browser types. Although our analysis and scenario focus on Android, as we will show at the end of this section, these concepts can be applied to other operating systems such as iOS.

Inter-App Communication Mechanisms

To perform the design of an IdM protocol in a secure way it is essential to choose the right communication channels between applications.

In Android, the communication between apps is performed through *Intent*, an inter process communication mechanism that can be of two types: *explicit*, when the sender specifies the name of the receiver, or *implicit*, whereby the sender only specifies the required task without specifying any receiver. In the design of IdM protocols, we suggest the use of explicit intent — to avoid known privilege escalation attacks compromising implicit intents [40]. Under the assumption of Android OS not rooted or compromised, the combination of explicit intents with the `startActivityForResult` method, which allows an app to display an Activity (basically a user interface) of another app and get a result back, guarantees that the message exchanged between two apps is confidential to the receiver app and the resulting message is authentic for the sender.

Whether it is required a communication between a browser and an application, Android also supports a communication via URI, allowing apps to register a URI with the OS. When the system browser attempts to access the URI (using the method `UIApplication.shared.openURL()`), the app that registered it is launched. Actually, this mechanism can be used also as an alternative for the `startActivityForResult` method, allowing an app to launch another app exchanging data as parameters of the URI. Android offers two modalities, well-defined in [93]:

App-declared Custom URI scheme (or *private-use URI scheme*): "Private-use URI scheme defined by the app and registered with the operating system. URI requests to such schemes trigger the app which registered it to be launched to handle the request."

App-claimed HTTPS URI scheme: "Some platforms allow apps to claim a HTTPS scheme URI after proving ownership of the domain name. URIs claimed in such a way are then opened in the app instead of the browser." In Android (6.0+) this mechanism is called *App Links* [25].

The developer to register an HTTPS URI scheme has to perform the following phases: configure the application to support HTTPS URI scheme, and publish on her website a JSON file with the URLs that the app manages. This file is used to provide domain verification.

Both types of redirect URIs are registered in the app's manifest by adding an intent-filter that specifies the URI value in the `<data android:scheme>` field.

About the App-declared Custom URI scheme, a known limitation is that the registration of a specific URI is not unique. Thus, a malicious app can register to the OS the same URI of another app. This results in showing a dialog-box to *User*, who has to choose with which application finishing the task. This problem is solved using App-claimed HTTPS URI scheme. In this case, the OS checks, after the installation of the app, that the app has the right to claim a specific URL, in this way checking the app identity.

Mechanisms for Establishing Federation

Another crucial aspect in the design of an IdM solution is the respect of the federation among the involved entities.

In Android, a way to check the identity of the developer of an app is using the Android method `getPackageInfo(client packageName, Package Manager.GET SIGNATURES)`. Considering a *SP* app sending a login request to an IdP_S through the *UA* app, with this Android method the *UA* app can extract the information about the certificate fingerprint included in the package of the *SP* app and then send this value (called *key_hash*) as a parameter of the login request. In this way, IdP_S is able to check the identity of the *SP* developer and the belonging to the federation.

Browser Types

Finally, another important security aspect is the selection of the right *UA*. As described in Sect. 2.2, a *UA* is used to perform the authentication process between SP_{app} and IdP_S, and could be a browser or a native application. If *UA* is a native application released by a trusted entity (e.g., the IdP_S), then we can trust the application to know the user's credential and authentication factors. If *UA* is a browser, we can choose different types of browser (definitions from [93]):

Embedded: "A user-agent hosted inside the native app itself (such as via a web-view), with which the app has control over to the extent it is capable of accessing the cookie storage and/or modifying the page content". The use of this type of browser is discouraged as there is a loss of isolation between SP_{app} and *UA*. If SP_{app} is malicious, then it can steal user's credentials or change authorization permissions. An additional limitation for the embedded browser solution is that it does not provide a SSO experience. Indeed, if the browser is integrated within the app, then the login session information is stored in (and only accessible to) the app and it is therefore not available to other apps. This forces users to re-enter credentials even if they have active login sessions with an *IdP*.

External: "A user-agent capable of handling the authorization request that is a separate entity or security domain to the native app making the request (such as a browser), such that the app cannot access the cookie storage, nor inspect or modify page content". It can be the browser of the operating system or an *in-app browser* that is "a programmatic instantiation of the browser that is displayed inside a host app, but retains the full security properties and authentication state of the browser" (called *Chrome Custom Tab* in Android). When it is

supported, in-app browser is suggested for usability reason.

Mapping between Android and iOS

In iOS, inter-app communications are performed via URI. It supports both App-declared Custom URI scheme and App-claimed HTTPS URI scheme, which in iOS (9+) is called *Universal Links* [77]. The custom-URIs are declared adding the `CFBundleURLTypes key` into the `Info.plist` file [76], while to support universal links, the developer has to update the app delegate to respond appropriately (more details in [77]).

About establishing a federation, a value equivalent to *key_hash* is the *Bundle ID*. As explained in the official documentation [75], *Bundle ID* identifies an application in a unique way, and can be used to validate the signature of an application. In an Xcode project, this value is stored in the information property list file (`Info.plist`). This file is then copied into the bundle of the application when the project is built. The main difference between an Android *key_hash* and an iOS *Bundle ID* is the association. The *key_hash* value identifies a developer of one or more apps, while the *Bundle ID* value is linked to a specific application.

Finally, regarding browser types, iOS supports both embedded and external browser. The *in-app browser* is called *SFSafariViewController* in iOS.

4.2 Facebook Social Login

At the time of our analysis, IdM protocols for mobile applications were available only in draft versions and, in general, their designs were based on traditional web technology rather than mobile devices. For these reasons, as first step of our work, we have extracted a rational reconstruction of Facebook SSO flow for native mobile applications. We have chosen Face-

book as it was the IdP more used for social native SSO [59].

Facebook (FB, for short) exploits a proprietary OAuth-based solution for both authentication and authorization. In this section, we will focus on the authentication process for Android native apps,[2] which allows app developers to integrate the FB login feature, and so exploit the FB native SSO solution. In Sect. 4.2.1, we describe the different FB native SSO solutions identifying the scenario that we will take into account during our security analysis. Sect. 4.2.2 describes our rational reconstruction of FB native SSO. Together with the details of the flow, we make a comparison with OAuth flows.

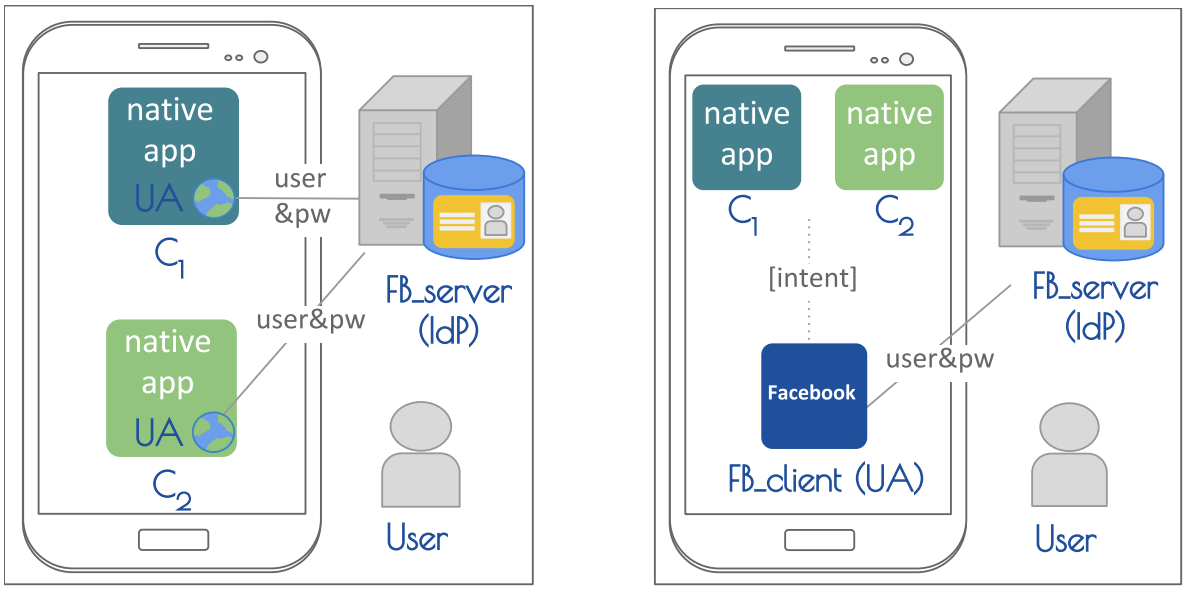
4.2.1 Facebook Native SSO Solutions

FB provides a Software Development Kit (SDK) to help mobile app developers to implement the login feature. Let us consider the situation in which there are two client applications (C_1 and C_2) that need to authenticate *User* with *FB_server*. As reported in the FB login guide [10],[3] the SDK has two different kinds of login implementation where *UA* is instantiated in different ways as shown in Figure 4.4: in case (a) C_1 and C_2 use an embedded browser, while in case (b) they use a dedicated native app (*FB_client*).

As pointed out in [100] and [13], if an embedded browser is used (case of Figure 4.4(a)), an app has full control of the hosted *UA*. Thus, if an app is malicious, then it can steal user credentials or change authorization permissions. An additional limitation for the embedded browser solution is that being the browser integrated within the app, the login session information is saved in (and only accessible to) the app and it is therefore not

[2]We focus on Android, even if the same concepts can be applied to other operating systems such as iOS (as explained in Sect. 4.1.2).

[3]In this paragraph, we refer to the login guide that we have analyzed at the time of our rational reconstruction. In the current login guide, the use of embedded browser is not more supported.



(a) Native Apps and Embedded Browsers.

(b) Native Apps Only.

Figure 4.4: Facebook Native SSO Solutions.

available to other apps. This forces the user to re-enter credentials even if she has active login sessions with an *IdP*. We thus disregard this solution and take into account the solution shown in Figure 4.4(b).

Facebook SDK Versions

Our rational reconstruction of FB solution was performed on Android 4.3.1, FB Android SDK 3.22.0 and FB Android PacKage (APK) 23.0.0.22.14. We also analyzed the FB flow with more recent versions (Android 5.1.1, FB Android SDK 4.1.0 and FB APK 44.0.0.26.142). Given that the only differences consist of some renaming and the addition/removal of some parameters that are not relevant for security, our description can be considered representative also for them. By contrast, a new way to manage the login process was added with version 4.11.0 (April, 2016). FB has recognized the security problems related to the use of an embedded browser. In the new versions, when the FB app is not installed in the smartphone of

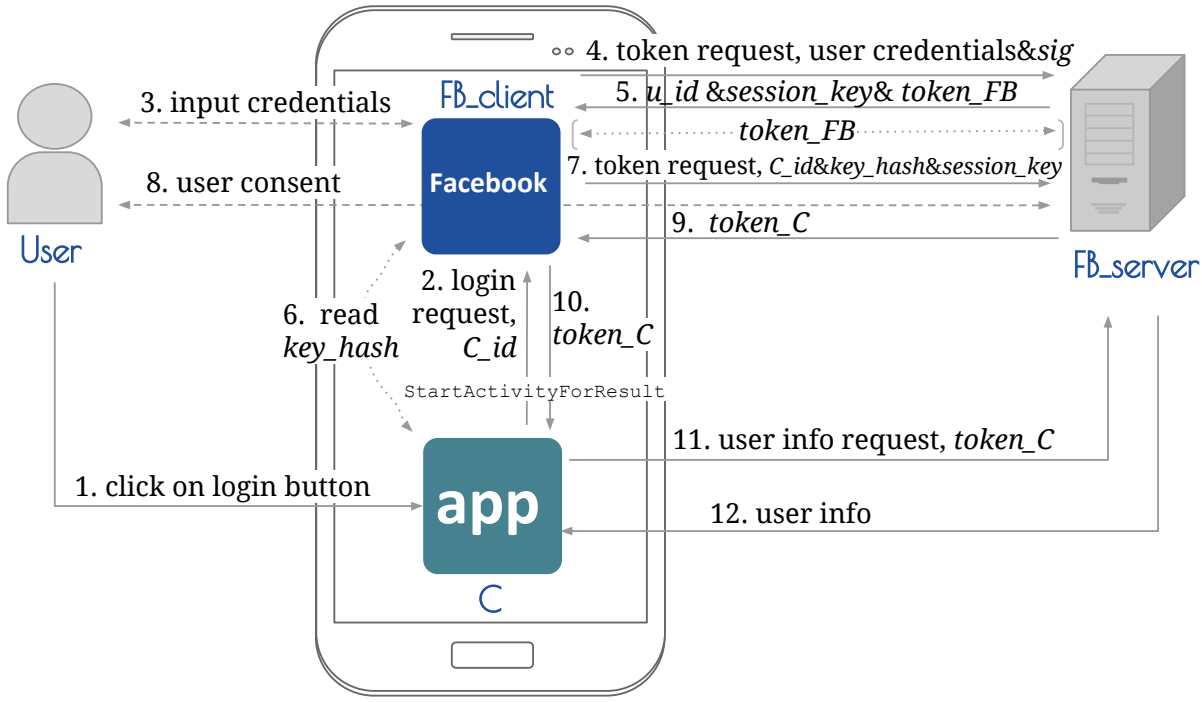


Figure 4.5: Facebook Native SSO Flow.

the user, the FB SDK presents the login dialog in a Chrome Custom Tab instead of a WebView. As described in Sect. 4.1.2, a Chrome Custom Tab is a full-page browser that is displayed inside a host app, but compared to a WebView, it retains the full security properties and authentication states of the system browser.

4.2.2 Rational Reconstruction

To reconstruct the FB native SSO solution, we implemented an application that integrates the FB Login feature following the FB documentation [10] and using the FB Android SDK, so from now on we will consider only one client (C app).

To obtain a precise description of the FB native SSO flow, we have used the following methodology: we have analyzed the source code of the FB SDK to understand the interaction between a C app and the FB_client in-

side the smartphone, and we have used the Fiddler proxy tool[4] to carefully inspect the HTTP(S) traffic between *FB_client* and *FB_server*.

Registration Phase. To use the FB login solution, C app developers have to register with FB. In this phase, C app developers have to enter the app package name and the certificate fingerprint (called, *key_hash*) of their app. The *key_hash* is a digest (SHA1) of the file CERT.RSA, that contains the public key of the developer, the signature of the app package derived from the private key of the developer, and other information about the certificate. After the registration phase, an identifier (*C_id*) is associated with the native app.

SSO Phase. Figure 4.5 depicts the most important steps, abstracting away the steps that are irrelevant for the authentication process. In Step 1, *User* opens C and clicks the "Login with Facebook" button to authenticate in the app. This triggers the invocation (Step 2) of an Activity from *FB_client* using the method `startActivityForResult`. The C app is put on hold for the Activity to return a result. In Step 3, *FB_client* presents to *User* an interface for prompting her credentials. *User* enters her credentials and activates a "Log in" button. In Step 4, *FB_client* interacts with *FB_server* in order to authenticate *User*. The HTTPS request contains the user credentials and a parameter (*sig*) that is the hash of the parameters of the request calculated using a specific value that is written in the *FB_client* app code. As a response to the authentication request (Step 5), *FB_client* gets the user information (*u_id*), among which there is a session value (*session_key*) and an access token (called *token_FB*). The *session_key* is used to set a session cookie linked to the user identity, while the *token_FB* is used by *FB_client* to obtain

[4]Available at `http://fiddler2.com/`.

configuration and other user information. The *token_FB* has no expiration date, but can be invalidated by the user either by changing password or logging-out from the device. In Step 6, using the Android method `getPackageInfo(client_packageName, Package Manager.GET_SIGNATURES)`, *FB_client* extracts the information about the certificate fingerprint (*key_hash*) included in the package of the *C* app. It is important to mention that the *key_hash* value is not sent by the *C* app, but is read by *FB_client* invoking an Android method. This means that the *C* app authentication is not performed using a value (e.g., *C_id*) that can be spoofed, rather, facilities for reliable authentication provided by the Android system are used. In Step 7, *FB_client* sends a token request for the *C* app to *FB_server*, including the *C_id*, *key_hash* and *session_key*. First, *FB_server* checks whether the provided *key_hash* matches the one previously entered by the developer for the given *C_id* in the registration phase. If there is no match then the flow is interrupted with an error, otherwise *FB_server* checks whether *User* authorized the *C* app. If the *C* app was not authorized by *User*, the response (Step 8) contains a consent form. The consent form asks *User* to grant or deny the *C* app to read the profile information. If the *C* app was already authorized by the user, the consent phase is skipped. If the user agrees, the response (Step 9) contains the *access_token* for the *C* app (*token_C*). In Step 10, *FB_client* returns control to the *C* app and provides *token_C* as result of the invoked Activity (see Step 2). Finally, the *C* app sends a request (Step 11) to *FB_server* including *token_C* to obtain the user information. The *C* app obtains the user information from FB and this proves that the app is under control of the corresponding user.

Comparison with OAuth Flows

We now show how the FB solution can be obtained as the combination of two OAuth flows. First, we can observe that Steps 3-5 of Figure 4.5

correspond to the steps of the Resource Owner Password Credentials flow with *User*, *FB_client* and *FB_server* playing the role of OAuth *RO* (Resource Owner), *C* (Client) and *AS* (Authorization Server), respectively. As the RO Password Credentials flow, the FB flow eliminates the need for storing the user credentials for future use, by exchanging the credentials with a long-lived access token (*token_FB*). Second, we have that Steps 2 and 7-10 of Figure 4.5 can be compared with Steps 1-3 and 6-7 of Figure 2.4(a) with *User*, *C*, *FB_client*, and *FB_server* playing the role of OAuth *RO*, *C*, *UA* (User Agent), and *AS*, respectively. We can observe that in the FB solution the WHCR (Web-Hosted Client Resource) entity is missing. The WHCR is the server side of *C* and is needed to give to *UA* the instructions to interpret the fragment values (e.g., access token value) returned in Step 3 and pass them to *C*. This procedure has been designed to manage the *same-origin policy* when *Cs* are web browser applications. Many native apps do not have a corresponding server, so in the FB native solution, the redirection URI is the same for every native app and has the value `fbconnect://success`. Indeed, it is the *FB_client* that receives the script and reads the access token out of the URI. In addition, we observe that Step 8 of Figure 4.5 corresponds to Step 2 of Figure 2.4(a) without the user login (the user identity is passed using the *session_key* parameter).

It is important to notice that, even if this parallelism is possible, the FB solution achieves an important feature compared to the OAuth flows: it authenticates *C*. In addition, we can observe that this solution manages a SSO experience: a *User* has to register only with *FB_server* and can use her FB credentials with different native apps (*Cs*); using *FB_client*, *User* enters her FB credentials just once, and the app will store the authenticated session. Every time a native app requires to login with *FB_server*, *FB_client* will employ the stored session without asking the user for credentials again.

Chapter 5

Phases 1-2: Application Context and *mID(OTP)*

In this chapter, we will describe the first two phases of our methodology. Sect. 5.1 will deal with the first phase, where an IdM designer is required to fill the application context (*AppCtx*) table (cf. Figure 4.2) for the application scenario under consideration, specifying the involved entities, the user agent (UA) choice, the data nature processed by the scenario and the support of specific authentication aspects (e.g., SSO and MFA). Sect. 5.2 will describe the second phase and detail our reference model, called *mID(OTP)*, for mobile IdM solutions. In Sect. 5.2.1, we will detail the *mID(OTP)* flow in two variants: *mID(OTP) Var1*, based on the Facebook solution, with a native application as *UA*, and *mID(OTP) Var2*, based on OAuth, with an external browser as *UA*. In Sect. 5.2.2, we will identify the security assumptions for *mID(OTP)* in both variants (native app or browser) and argue that they allow to prevent violations of the security goals that we have identified in Sect. 5.2.3. Finally, in Sect. 5.2.4, we will compare *mID(OTP)* to Facebook and OAuth for native apps solutions.

5.1 Phase 1: Application Context Identification

In the first phase of our methodology, the IdM designer is required to fill the *AppCtx* Table of Figure 4.2. In general, the information specified could be manifold; we require at least these categories:

Entities. The involved entities are obtained after an analysis of the overall architecture. To clarify the role of each entity, it is required a mapping between the roles usually involved in an IdM solution and the entities of the analyzed solution. Thus, we have defined a role mapping function rf that associates to each $r \in IdMRoles$ an entity $e \in Entities$. *IdMRoles* is the set of roles described in Sect. 2.1, namely *IdMRoles* = $\{User,\ SP_{app},\ SP_S,\ UA,\ IdP_S,\ AP,\ IdB,\ TP_{app},\ TP_S,\ HWToken,\ AS,\ RS\}$. As not each scenario considers authorization or MFA, we have to introduce an entity, called *Null*, that is always implicitly included in the *Entities* set. If a role r is not played by any entity of the scenario, then $rf(r) = Null$.

UA Choices. In the mobile context, two possible design choices are available: a *UA* could be played either by a browser or by a native app. We will define the MSC and security assumptions for both the variants in Sect. 5.2:

- *mID(OTP) Var1* with a native application as *UA* based on the Facebook solution, and
- *mID(OTP) Var2* with an external browser as *UA* based on the OAuth for native app proposal.

The *UA* choice depends mainly on the authentication protocol used. If it requires the installation of ad-hoc mobile application (e.g., for generating a OTP) then the *UA* will be probably played by this appli-

cation. Otherwise, for authentication protocol that are browser-based (e.g., SAML 2.0) the solution will use a browser.

Data Nature. The data nature row specifies which kind of data the solution will treat, that could be one or more among *anonymous*, *personal*, and *sensitive*, where

- *anonymous data* are "any data that cannot be associated to any identified or identifiable data subject" [57, §4, lett. n];
- *personal data* are "any information relating to an identified or identifiable natural person ('data subject'); an identifiable person is one who can be identified directly or indirectly [...] by an identification number or one or more factors specific to his physical, physiological, mental, economic, cultural or social identity" [48, §2, lett. a]; and
- *sensitive data* are "any data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, trade-union membership, and the processing of data concerning health or sex life" [48, §8].

Authentication Aspects. Our methodology presents a list of authentication aspects (as shown in Figure 4.2) that the IdM designer has to select taking into account the security and usability level that the analyzed solution must achieve. Regarding authentication — which is the process of verifying that an entity is the entity that it claims to be — our methodology considers:

Multi-Factor Authentication (MFA) support. As described in Sect. 2.1.1, a MFA augments the security of a single-factor authentication by combining two or more authentication elements (factors) of different categories (e.g., a password combined with a code sent to a

mobile device, or some biometric data). The mutual independence between authentication factors is essential, otherwise a compromised factor could invalidate the security of the protocol.

Session handling. If a *User* has already a login session with an *IdP*, then she can access new *SP* apps without re-entering her IdP credentials; only the user consent is required. This requirement improves at the same time the user's experience (usability) and the security (e.g., stronger password selection).

OTP choice. As we will describe later, *mID(OTP)* is parametric on the OTP generation (i.e. it supports different OTP generation approaches). In our analysis, we have detailed the OTP generation box for TOTP and Challenge-Response (CR) approaches (described in Sect. 2.1.1). If the application scenario requires an OTP approach different from TOTP or CR, in the next phase the IdM designer will (manually) design the corresponding OTP box.

The choice of the right authentication aspects subsumes an analysis of the legal aspects (cf. Chapter 3) in force in the countries where the application is going to be deployed. For example, if the data processed by the application are sensitive data, then a MFA solution is required. In Table 5.1, for the Italian case, we have linked the Italian national law on

Table 5.1: AuthN Aspects and IdM/Privacy Italian Laws.

AuthN Aspect	**IdM/Privacy Italian Laws**
MFA	SPID (Annex 4)
	Personal Data Protection Code (Section 31, 33 and 34)
SSO	SPID (Annex 3)
Multi-IdPs	8 *IdPs* are accredited as SPID *IdPs*
Cross-border	SPID as national eID notified to EU (eIDAS, Article 9)
Global logout	SPID (Annex 3)

digital identities (SPID [9]) and the privacy code (Personal Data Protection Code [57]) with the authentication aspects described before and in the following section. In addition, for the *AppCtx* identification of the four real-world scenario described in Chapter 7, we have referred to the decisional diagram described in Sect. 3.2.3.

Extending Authentication Aspects

In the definition of *mID(OTP)* (cf. Sect. 5.2) we will consider a SSO experience with optionally MFA and session handling. For different scenarios, as shown in the real-world scenarios described in Sect. 7, our design is easily extendible to more complex identity federation aspects, such as:

Multi-IdPs support. In this case, the *SP* app automatically supports the login with different *IdPs* having a federation with an external entity that manages the integrations with different *IdPs*.

Cross-border support. In this case, there is the additional requirement that *IdP*s supported by the *SP* apps belong to other countries. We will focus on European cross-border authentication.

Global logout support. In this case, *IdP* and all the *SP* apps used by *User* must delete their user sessions. In this way, at the following access to any of the *SP* apps an authentication process with *IdP* is required.

In addition, our methodology can be extended to cover different authorization aspects. Authorization is the process of determining which actions an authenticated entity can perform. For authorization, our methodology can easily support:

Permission model at Authorization Server (AS). This requires a mechanism composed of two steps: one in which *SP* apps can request access to

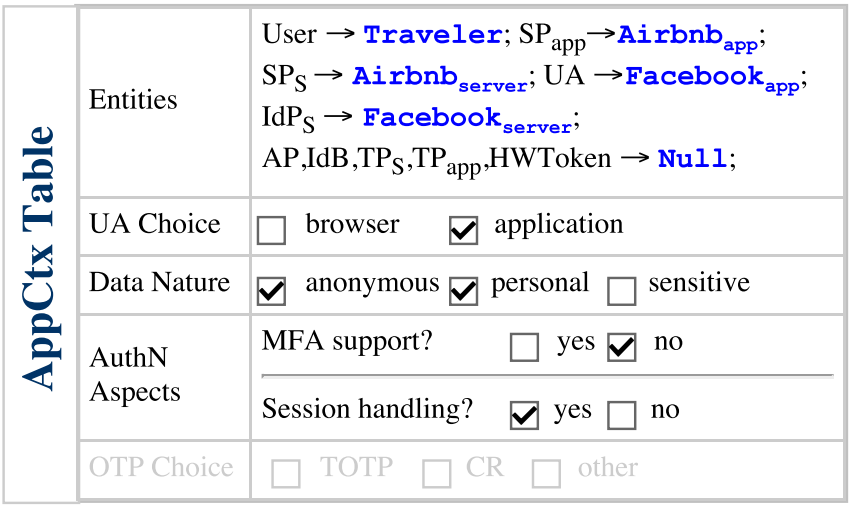


Figure 5.1: Facebook *AppCtx* Table.

selected pieces of information, and one in which *Users* instruct *AS* to grant (or deny) requested accesses.

Fine-grained access control. With a fine-grained access control, *User* can choose which resources and attributes share with *SP* in a selected way — extending the classic "all-or-nothing" approach.

Similarly to identity federation, also in the authorization context can be established a federation, defined as an agreement between two or more domain specifying how access policies will be exchanged and managed for cross-domain authorization purposes. For access federation, our methodology can support:

Cross-domain access control. *AS* from domain *D1* can release an access token that can be used to access resources in *RS* (Resource Server) of domain *D2*.

5.1.1 Facebook Example

To provide an example of filling an *AppCtx* table, let us consider the Facebook sign-in solution. To determine the authentication aspects, an in-

formal description of the solution is required. Facebook solution can be described as "a solution that using the Facebook app permits *Users* to login into different mobile applications with Facebook credentials (username and password) with a SSO experience, and delegate the access of their Facebook resource". Being a social network solution, the user data will be mostly anonymous data (e.g., number of friends and followed pages) and some personal data (e.g., the name, address, age and so on). About authentication aspects, Facebook does not support a MFA and the session is handled to provide SSO for different applications. Together with authentication, Facebook also considers the access delegation aspect. Considering the integration of Facebook login solution with Airbnb, a mobile app for renting short-term lodging, we have the following set of entities:

$Entities_{FB} = \{Traveler, Airbnb_{app}, Airbnb_{server}, Facebook_{app}, Facebook_{server}\}$

that, using the entity mapping function rf, are the images of the following roles $User \overset{rf}{\longmapsto} Traveler$; $SP_{app} \overset{rf}{\longmapsto} Airbnb_{app}$; $SP_S \overset{rf}{\longmapsto} Airbnb_{server}$; $UA \overset{rf}{\longmapsto} Facebook_{app}$; IdP_S,AS,$RS \overset{rf}{\longmapsto} Facebook_{server}$; AP,IdB,TP_S,TP_{app},$HWToken \overset{rf}{\longmapsto} Null$. Thus, as shown in Figure 5.1, the UA choice for FB solution is an application.

5.2 Phase 2: Customization of *mID(OTP)*

The second phase of our methodology takes as input the *AppCtx* table, and, based on the *AppCtx* values, the reference model *mID(OTP)* is almost automatically customized generating as output the MSC (flow description), the security assumptions, and security goals for the application scenario. In particular, aspects that determine different outputs are:

UA choice: as described in Sect. 5.1, *UA* could be played either by a browser or by a native app. As we will describe in Sect. 5.2.1 and Sect. 5.2.2, this choice varies the MSC and assumption selection.

MFA support: this aspect varies the security goal selection. As we will describe in Sect. 5.2.3, basic authentication and MFA requires different goals.

OTP choice: mID(OTP) is parametric on the OTP generation (i.e. it supports different OTP generation approaches). In Chapter 7, we will detail: the OTP box for a TOTP approach using a native app to generate OTPs; and a Challenge-Response (CR) approach using a native app that exchanges a challenge (through the NFC technology) with a smartcard that provides the cryptographic primitives for signing the challenge and returning a response. If the application scenario requires an OTP generation approach different from these approaches, then some extra design is required.

In the following, we will describe *mID(OTP)* detailing the different choices for a correct customization. In Sect. 5.2.1, we will describe the two MSC variants:

- *mID(OTP) Var1* with a native application as *UA*, based on the Facebook solution; and

- *mID(OTP) Var2* with an external browser as *UA*, based on the OAuth for native app proposal.

In Sect. 5.2.2, we identify the security assumptions for *mID(OTP)* in both variants (*UA* native app or browser) and argue that they allow us to prevent a violation of the security goals that we have identified in Sect. 5.2.3. Finally, in Sect. 5.2.4, we compare *mID(OTP)* with the Facebook native SSO solution and OAuth.

5.2.1 Message Sequence Charts of *mID(OTP)*

mID(OTP) involves the IdM roles described in Sect. 2.2: the human participant (*User*), the server that authenticates the user (IdP_S), the entity that manages authentication and token exchange (*UA*), the native app that uses the authentication assertions of the IdP (SP_{app}), and (optionally) the app and server to manage MFA (TP_{app} and TP_S). *mID(OTP)* consists of three phases: *registration*, *activation* and *exploitation*, which we describe in the following subsections for the following two variants:

Var1. Starting from the FB native SSO solution (cf. Sect. 4.2) and taking advantage of our security analysis (cf. Sect. 6.1.3), we have derived a reference model for native IdM solutions with a native application as *UA*. As *UA* is involved in the authentication phase with IdP_S, it must be assumed trusted in knowing the user's IdP credentials. Thus, we assume that it is released directly by IdP_S and is called *IDOTP* (the entity-role mapping between the entities of *mID(OTP)* and the role of an IdM system are summarized in Table 5.2). This model generalizes the one proposed by FB, in such a way that it can be used as a reference model by any IdP willing to provide its own IdM solution and (optionally) to support a MFA, based on the generation of OTPs. In the MFA case, *IDOTP* also plays the role of TP_{app}.

Var2. Starting from OAuth for native app (cf. Sect. 2.2.4), we have derived a reference model with an external browser as *UA*, indicated with *Browser*. As a native app can be (easily) extended to support the generation of an authentication factor (e.g., by adding the code for a OTP generator or a library to process the user's fingerprint), together with a browser, to support MFA we consider a native app (called *OTPApp*) that plays the role of TP_{app}. This model generalizes the one proposed by OAuth working group, in such a way that it can

be used as a reference model for any protocol that uses a browser for redirections (e.g, SAML), and (optionally) for supporting a MFA based on the generation of OTPs.

In both variants, TP_{app} (either *IDOTP* or *OTPApp*) could generate OTPs either directly (e.g., using a TOTP approach) or requiring the interaction with an external *HWToken* (e.g., a smartcard). To simplify the description of the exploitation phase and maintain the generality with respect to the OTP approach, in this section we consider only TP_{app} (*IDOTP* or *OTPApp*), abstracting away the interaction with a *HWToken* (we will provide details in Chapter 7 for the different scenarios). In addition, we focus on scenarios where IdP_S and TP_S are played by the same entity that we called *IdTP*. In general, the token validation can be performed by entities belonging to different security domains (this requires extra design and we will consider as future work).

Registration and Activation Phases of *mID(OTP)*

The registration phase of *mID(OTP)* is performed by the SP_{app} developers and consists of the exchange of some information (called *metadata*) about SP_{app}. The metadata required by *IdTP* can vary, however at least the following two values must be always present:

Var1. The app package name and the certificate fingerprint (called, *key_hash*) of the app. The *key_hash* is a digest (SHA1) of the file CERT.RSA,

Table 5.2: Mapping *mID(OTP)* Entities and IdM Roles.

***mID(OTP)* Entity**	**IdM Role**
IDOTP	*UA* and TP_{app}
OTPApp	TP_{app}
IdTP	IdP_S and TP_S

that contains the public key of the developer, the signature of the app package calculated using the private key of the developer, and other information about the certificate. Note that *key_hash* depends on the private key of the SP_{app} developer and it is thus different for apps implemented by different developers. After the registration phase, *IdTP* associates an identifier with the registered SP_{app} app.

Var2. The app package name and the redirection uri of the application (called, *redirect_uri*). A *redirect_uri* corresponds to the HTTPS scheme URI claimed by SP_{app} to manage the redirection inside the smartphone (cf. Sect. 4.1.2).

The registration phase can be performed in different ways, e.g., entering the data into an online dashboard or via an email exchange. As a trust relationship between SP_{app} and *IdTP* is established as result of the registration phase, it is important that *IdTP* validates the SP_{app} data and in some cases (e.g., when user personal or sensitive data are involved) a service-level agreement could be required as well.

The activation phase of *mID(OTP)* is performed by the *User* to establish an authentication session with *UA* on her smartphone — for *Var1*, *IDOTP* obtains a token called *token_IdP*; for *Var2*, *Browser* obtains a cookie called *session_cookie* — and (optionally) to configure TP_{app} (*IDOTP* or *OTPApp*) to support a second authentication factor as well (e.g., requiring the creation of a PIN code). As this phase depends strictly on the IdM entities involved, we will detail it in the context of the real-world scenarios of Chapter 7.

Exploitation Phase of *mID(OTP)*

The exploitation phase of *mID(OTP)* is performed every time *User* accesses a SP_{app} that requires the SSO experience.

The steps for *Var1* are shown in Figure 5.2 and consist of:

S1. *User* opens SP_{app}.

S2. SP_{app} sends a request to SP_S including a token session *token_sync*.

S3. SP_S checks the validity of *token_sync*. If *token_sync* has expired, SP_S sends an error message asking for a login to SP_{app}, otherwise Step S7 is executed.

A1. *User* clicks the login button.

A2. SP_{app} sends a token request to *IDOTP*.

A3. *IDOTP* reads the *key_hash* value of SP_{app}.

A4. *IDOTP* sends a token request to *IdTP* including the *key_hash* value of the SP_{app} app and *token_IdP*. *key_hash* is used by *IdTP* to validate the SP_{app} identity and *token_IdP* corresponds to the user credentials entered during the activation phase.

A5. If SP_{app} and *token_IdP* are valid, *IdTP* returns to *IDOTP* a consent containing the meta-data of SP_{app}. After user consent, optionally, an OTP is generated following one of the algorithms described in Sect. 2.1.1.

A6. If *User* agrees and (optionally) the authentication succeeds, *IdTP* returns a token (*token_SP*) to the *SP* app. *token_SP* contains the identity of *User*, *IdTP* and SP_{app}, and is digitally signed with $K^{-1}_{IdP_S}$, the private key of *IdTP*.

A7. *IDOTP* returns *token_SP* to SP_{app} as result of Step A2.

S4. SP_{app} sends a token request to SP_S including *token_SP*.

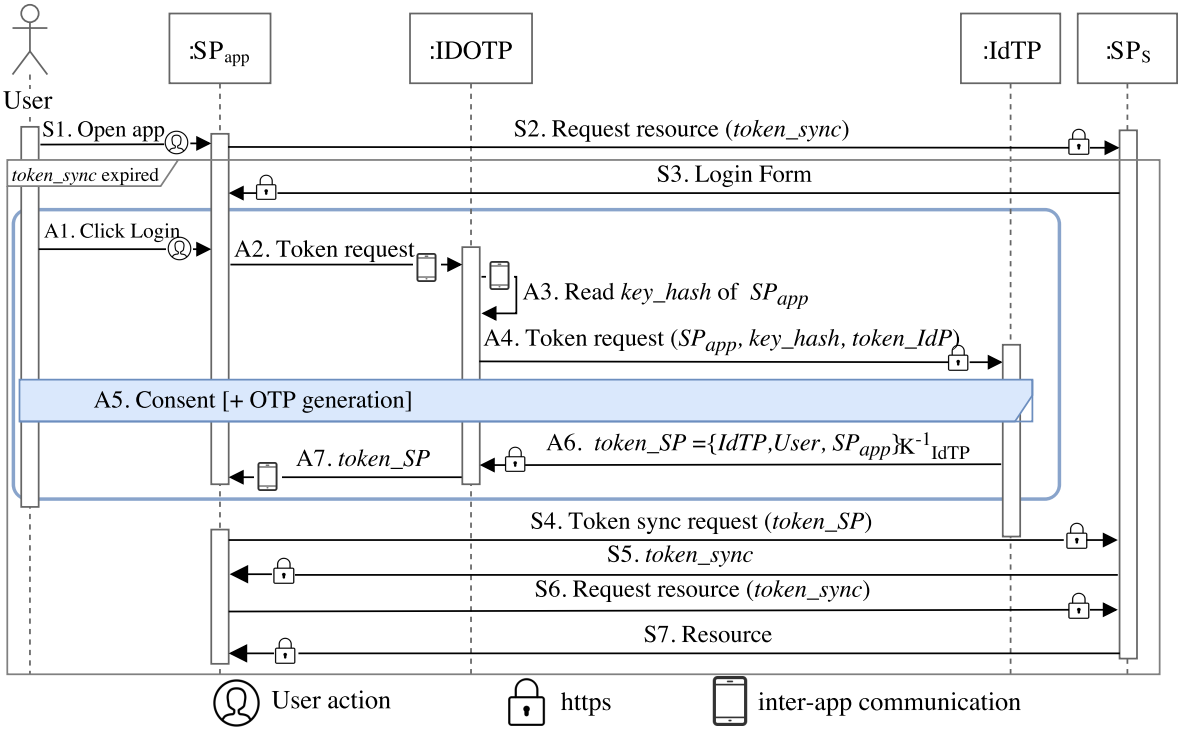


Figure 5.2: *mID(OTP)* Flow with a Native App as UA.

S5. SP_S checks the validity of *token_SP*, and if it is valid, SP_S creates and sends to SP_{app} a token (*token_sync*). This token will be used by SP_{app} to synchronize user data in the future interactions, until its expiration.

S6. When SP_{app} needs to synchronize data, SP_{app} sends a request to SP_S including *token_sync*.

S7. SP_S checks *token_sync* and, if it is valid, sends the requested resource to SP_{app}.

We have labeled the steps with "S" and "A". S steps are related to the *SP* (but note that the steps we have considered are only an example and each *SP* could support different solutions). A steps represent the steps related to the authentication solution. As the S steps can vary depending on the choices of the SP developers, in our analysis, we will focus on the A steps.

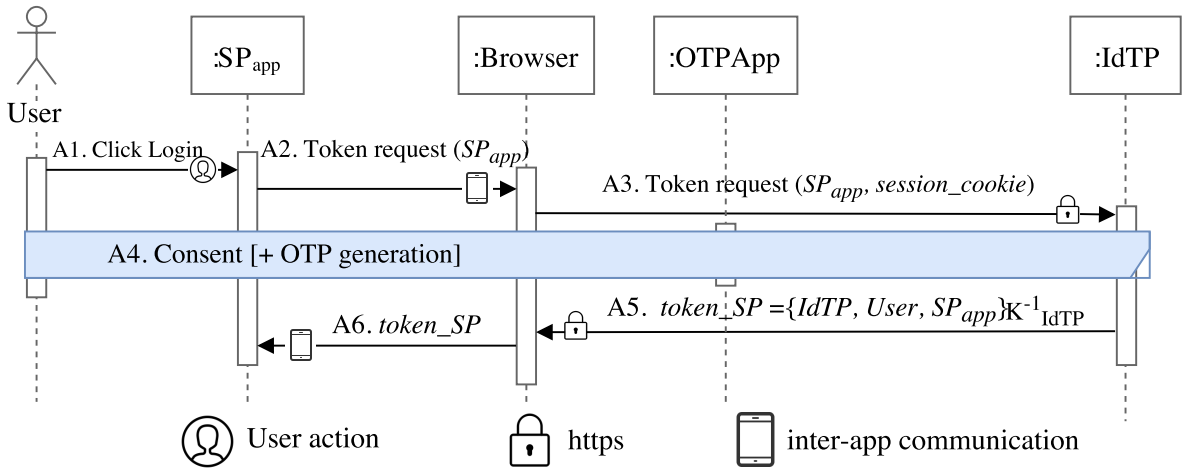


Figure 5.3: *mID(OTP)* Flow with an External Browser as UA.

The A steps for *Var2* are shown in Figure 5.3. In general, the authentication with *IdTP* can be performed in this phase. However, being that the process starts with a *SP* app, *User* cannot be sure that the login form is authentic (SP_{app} can fake a login browser page). One way that *User* has to avoid phishing attack — as suggested in [93, §8.7] — is to login directly in a trusted *Browser* and obtain a *session_cookie*. Thus, in our flow (Figure 5.3) we have assumed that *User* has an active session (*session_cookie*) with *IdTP* on *Browser*. In detail, these are the steps:

A1. *User* clicks the login button.

A2. SP_{app} opens *Browser* with a token request to *IdTP* passing as parameter its identifier.

A3. *Browser* follows the link redirection and sends the token request to *IdTP* including *session_cookie*, which corresponds to the user credentials entered previously in *Browser*.

A4. If SP_{app} and *session_cookie* are valid, *IdTP* returns to *Browser* a consent containing the meta-data of SP_{app}. After user consent, optionally, an OTP is generated following one of the algorithms described

in Sect. 2.1.1 and involving *OTPApp*.

A5. If *User* agrees and (optionally) the authentication succeeds, *IdTP* returns a token (*token_SP*) to SP_{app} following its *redirect_uri* (value stored by *IdTP* during the SP_{app} registration phase). *token_SP* contains the identity of *User*, *IdTP* and *SP*, and is digitally signed with $K^{-1}_{IdP_S}$, the private key of *IdTP*.

A6. *Browser* returns *token_SP* to SP_{app} following the redirection specified by *IdTP*.

5.2.2 Defining *mID(OTP)* Assumptions

In order to simplify the security analysis phase performed by the IdM designer, we have identified the security assumptions and the security goals of *mID(OTP)*. We consider the following assumption categories: *Background Assumptions*, *Trust Assumptions*, *Communication Assumptions*, *Activation Assumptions* and *User Behavior Assumptions*. In details:

Background Assumptions. For both *Var1* and *Var2*, user's device is subject to these assumptions:

(BA1) Integrity and confidentiality of data stored in the device, i.e. an app cannot read or modify data stored by another app.

(BA2) There is no surveillance software (e.g., keylogger) installed on the user's device capable of reading the values that *User* types.

Trust Assumptions. They clarify the trust relationships between the different entities. The trust assumption of *mID(OTP)* that we have identified for both *Var1* and *Var2* is:

(TA) *IdTP* is trusted by SP_{app} on identity assertions. That is *IdTP* releases only valid and correct identity assertions.

Communication Assumptions. They specify the concrete implementation of the communication channels required in *mID(OTP)*. For *Var1*, communications between the parties are subject to the following assumptions:

($CA1_{Var1}$) The communication between SP_{app} and *IDOTP* is carried over an inter-app communication implemented using `StartActivityForResult()`. This Android method — which allows an app to execute another app and get a result back — guarantees that the SP_{app} app that sends a request to *IDOTP* at Step A2 in Figure 5.2 is the same app that receives the result back from *IDOTP* at Step A10.

($CA2_{Var1}$) To read the *key_hash* value (Step A3 of Figure 5.2), *IDOTP* uses the Android method `getPackageInfo(client packageName, PackageManager.GET SIGNATURES)`, which extracts the information about the certificate fingerprint included in the package of SP_{app}.

($CA3_{Var1}$) The communication between *IDOTP* and *IdTP* occurs over a unilateral SSL or TLS channel (henceforth SSL/TLS), established through the exchange of a valid certificate (from *IdTP* to *IDOTP*).

For *Var2*, communications between the parties are subject to the following assumptions:

($CA1_{Var2}$) *Browser* to exchange *token_SP* with SP_{app} (Step A9 of Figure 5.3) uses Android App Links. This redirection scheme guarantees that *Browser* sends *token_SP* to the correct SP_{app} application (previously validated by the operating system as explained in Sect. 4.1.2).

($CA2_{Var2}$) The communication between *Browser* and *IdTP* occurs over a unilateral SSL or TLS channel (henceforth SSL/TLS), established

through the exchange of a valid certificate (from *IdTP* to *Browser*).

(CA3$_{Var2}$) If MFA is supported, *OTPApp* uses the Chrome Custom Tabs (cf. paragraph "Browser Type" of Sect. 4.1.2) to launch *Browser*.

These assumptions refer to a concrete implementation of the communication channels. It is important to point out that in Sect. 6.2.1 we will provide the formal counterpart abstracting away the implementation details. By doing so, any implementation satisfying the abstract assumptions can be used in place of the implementation mentioned above (e.g., considering a similar solution in the case of iOS), and the results of our security analysis still hold.

Activation Assumption. Phishing attacks (in this context we mean a malicious app that creates a fake login form and steals the user's credentials) are one of the most common types of attack and usually are beyond the scope of an authentication protocol. In our analysis, together with a secure communication, we assume that no phishing is possible during the activation phase of *mID(OTP)*:

(AA) The activation phase is correctly performed by *User*. That is, *User* downloads the correct *IDOTP* or *OTPApp* (i.e. they are not fake apps) and correctly follows the activation phase, and the communication channels that are involved in this phase are secure.

User Behavior Assumptions. To enforce a correct execution of the flow and to investigate the security consequences of a stolen smartphone, in our analysis we take into account the following behavioral rules. For *Var1*:

(UBA1$_{Var1}$) *User* enters her credentials and (optionally) values for the OTP generation only in the correct *IDOTP* app being careful not to be seen by other people.

(UBA2$_{Var1}$) *User* is the only person using the *IDOTP* app that has been activated with her identity.

For *Var2*:

(UBA1$_{Var2}$) *User* enters her credentials only in a trusted *Browser*, and (optionally) values for the OTP generation only in the correct *OTPApp*, being careful not to be seen by other people.

(UBA2$_{Var2}$) *User* is the only person using *Browser* (and if present, *OTPApp*) that is associated to her identity).

Note that, with UBA2$_{Var1}$ and UBA2$_{Var2}$ we are implicitly assuming that the user smartphone is not stolen.

5.2.3 Defining *mID(OTP)* Security Goals

mID(OTP) supports both a basic username-password authentication and a MFA solution using OTPs. Thus, preliminary to define the security goals expected by *mID(OTP)*, it is important to clarify the peculiarities of a MFA solution compared to a basic username-password authentication; in doing so, we introduce some concepts that will be the key for the formal analysis.

Basic Authentication in *mID(OTP)*

In a basic authentication, the expected security goal is:

(G1$_{BA}$) *SP* authenticates *User*

For the concept of *authentication* we refer to [82], where it is defined in the following way: whenever the entity B completes a run of the protocol apparently with the entity A, then (i) A has previously been running the

protocol apparently with B, and (ii) the two entities agree on a message M.

For $\mathrm{G1}_{BA}$ to hold, *User* is required to provide an authentication factor: either credentials (something only she knows) or a session token (e.g., a cookie stored in her browser) in order to be properly identified by *SP*. If this is the case, it is possible to specify a minimum set of security assumptions (e.g., on the behavior of *User* or on the communication channels) that are necessary to guarantee $\mathrm{G1}_{BA}$. For example, if the channel used for the login is not https, then an intruder can eavesdrop the *User*'s password and impersonate her in the future. For *mID(OTP)* as basic authentication, all the assumptions that we have identified in Sect. 5.2.2 are necessary to prevent a violation of $\mathrm{G1}_{BA}$ (an informal proof of this statement is reported in Appendix A).

Multi-Factor Authentication in *mID(OTP)*

A MFA solution augments the security of the basic username-password authentication by exploiting two or more authentication factors. Following the definition given in Sect. 2.1.1, we infer that *mID(OTP)* is a two-factor authentication solution using knowledge and ownership elements (factors). As the real-world scenarios that we will describe in Chapter 7 do not involve inherence elements, we consider their analysis as a future work. In addition, instead of considering the independent factors, we introduce the concept of instance-factors.

We call *instance-factor* (*IFactor*) every specific instance of either an ownership factor ($IFactor_o$) or a knowledge factor ($IFactor_k$). The MFA solution that we propose for *mID(OTP) Var1* contains three instance-factors:

- the $IFactor_o$ *token_IdP* that is stored in *IDOTP* and in *IdTP* as a result of the activation phase (used as a session token in place of the

user credentials to provide a SSO experience);

- another $IFactor_k$ that can vary according to the specific OTP generator used, e.g., a PIN known by the user (used to protect the OTP generator);

- an $IFactor_o$ that is stored in *IDOTP* (and possibly shared with *IdTP*), according to the OTP generator used (e.g., a seed value or a private key).

Note that the $IFactor_o$ *token_IdP* is present in all instances of our solution for *Var1*, whereas the other two factors may differ depending on the specific solution (and this is the reason why we cannot name them explicitly a priori). Equally, the MFA solution for *mID(OTP) Var2* has as $IFactor_o$ the *session_cookie* that is stored in *Browser* plus the instance factors linked to the OTP generation.

Compared to the usual notion of authentication factors, instance-factors can have a dependency. For example, the two $IFactor_o$ instance-factors are stored in *IDOTP*. Thus, by breaching the *IDOTP* app both of them are compromised. However, it is important to note that different mitigations can be implemented for the different instance-factors. For example, in our solution, if a *User* realizes that the *IDOTP* has been compromised (e.g., if her smartphone has been stolen), she can invalidate *token_IdP*, thus blocking possible attacks.

We are not aware of any formal definition of the MFA property apart from [31]. By generalizing the definition in [31] for any OTP generation approach and considering a solution involving n instance-factors, we can define the following security goal:

($G1_{MFA}$) Goal ($G1_{BA}$) (i.e. *SP* authenticates *User*) holds even if an intruder knows up to $n-1$ instance-factors.

Thus, the addition of instance-factors ensures some "redundancy", meaning that even if one of them is compromised there are no attacks.

Thanks to this redundancy, we can define two different assumption categories. We call *strong assumption* an assumption that is necessary to guarantee $\mathrm{G1}_{BA}$. Meaning that, whenever it is not valid (e.g., not implemented properly) causes the compromise of all the instance-factors or of the authentication assertion *token_SP*. We write SA_{Var1}, SA_{Var2} to denote the *set of all the strong assumptions* for *Var1* and *Var2*, respectively. We call *weak assumption* (*wa*) an assumption that, whenever it is not valid or not implemented properly, causes the compromise of a non-empty set of instance-factors of the same type, i.e. either $IFactor_o$ or $IFactor_k$. We refer to this set as the *set of instance-factors associated with wa* and denote it by writing $IF(wa)$.[1] For example, if a weak assumption *wa1* states that the intruder cannot read the values typed by *User*, and in the authentication process *User* has to enter her *password* and *PIN*, then $IF(wa1) = \{password, PIN\}$. This definition can be easily extended to a set of weak assumptions WA' as follows: $IF(WA') = \bigcup_{wa_i \in WA'} IF(wa_i)$. We write WA_{Var1}, WA_{Var2} to denote the *set of all the weak assumptions* for *Var1* and *Var2*, respectively.

Referring to the assumptions defined in Sect. 5.2.2, we have the following set of strong and weak assumptions of *mID(OTP)* as a MFA solution:

$$SA_{var} = \{\mathrm{TA},\ CA1_{var},\ CA2_{var},\ CA3_{var},\ \mathrm{AA}\}$$

$$WA_{var} = \{\mathrm{BA1},\ \mathrm{BA2},\ UBA1_{var},\ UBA2_{var}\}$$

where $var \in \{Var1, Var2\}$.

The notions that we just introduced allow us to rephrase the definition of the security goal ($\mathrm{G1}_{MFA}$) of a MFA solution in the following way:

[1] To compromise all instance-factors, at least two weak assumptions must be not valid.

($G1_{MFA}$) Goal ($G1_{BA}$) (i.e. *SP* authenticates *User*) holds under SA_{var} and under a chosen subset of weak assumptions (WA') such that the set of instance factors associated to $WA_{var} \setminus WA'$ does not include all the instance-factors.

where $var \in \{Var1, Var2\}$.

In $G1_{MFA}$, we considered (among others) the instance-factors linked to the OTP generation. In addition, as reported in Sect. 2.1.1, an OTP "should be non-reusable and non-replicable." Indeed, if the OTP is not fresh, then the knowledge of an OTP leads to the same attacks possible when knowing the instance-factors linked to its generation. Thus, it is crucial that the following security goal about the OTP is satisfied:

(G2) The OTP must prove its origin (meaning that *IdTP* authenticates TP_{app} or *HWToken*, as it is the only entity knowing a seed value shared with *IdTP* or possessing a private key), and it is non-reusable (*IdTP* accepts only one OTP for a specific operation avoiding replay attacks).

5.2.4 Comparisons with State-of-the-Art Solutions

In this section, we compare *mID(OTP)* with the Facebook native SSO solution and the OAuth specification for native apps, taking into account the security assumptions described in Sect. 5.2.2. We have assumed that the user smartphone is not stolen and rooted, the OS is not compromised, and IdP_S is trusted. Given that for all the solutions, we consider assumptions $UBA2_{Var1}$, $UBA2_{Var2}$, BA1, BA2, and TA as satisfied.

mID(OTP) *Var1* **vs Facebook Solution**

In Table 5.3, we compare the two solutions that use a native application as UA: Facebook solution and *mID(OTP) Var1*. As we can observe, $CA1_{Var1}$,

$CA2_{Var1}$, $CA3_{Var1}$ are satisfied by Facebook solution, thus SP_{app} is correctly authenticated and the result of the authentication process ($token\_SP$) will be returned only to the right SP_{app}. In the FB solution, the non-fulfillment of $UBA1_{Var1}$ causes the phishing attack: users are not able to identify the app that shows the login form. In addition, this causes the client impersonation attack: a malicious app, knowing the user credentials, is able to obtain a token that is targeted for the *FB_client* app. Finally, the use of bearer tokens makes the FB solution vulnerable to the user impersonation attack: a C app cannot locally verify that the received token was meant for itself and check the authenticity and integrity of it. Compared to the FB solution our solution is capable to fulfill the identified security assumptions. This mitigates the vulnerabilities (phishing, user impersonation and client impersonation) discovered in FB and analyzed in Sect. 6.1.3. For example, the activation phase of our reference model allows for better mitigation of phishing by design, as users directly interact with *IDOTP*. While FB solution requires a redirection from a (possible malicious) *SP* to the *FB_client* app, thus users can be cheated by a fake login invoked by SP.

mID(OTP) *Var2* **vs OAuth for Native Apps**

In Table 5.4, we compare the solutions that use an external browser as UA: *mID(OTP) Var2* and OAuth for native apps [93] (Custom URI and

Table 5.3: Comparison among Facebook Solution and *mID(OTP) Var1*.

Asm	**Facebook Solution** (Fig. 4.5)	***mID(OTP)*** *Var1* (Fig. 5.2)
$CA1_{Var1}$	✓	✓
$CA2_{Var1}$	✓	✓
$CA3_{Var1}$	✓	✓
AA	n.a	✓
$UBA1_{Var1}$	×	✓

HTTPS URI). As we can observe:

Custom URI. The non-fulfillment of UBA1$_{Var1}$, as in FB, causes the phishing attack; while the non-fulfillment of CA1$_{Var2}$ permits a malicious C app to perform a run of the protocol without being recognized.

HTTPS URI. CA1$_{Var2}$ is correctly implemented using the HTTPS URI redirection schema. While, CA2$_{Var2}$ is satisfied using TLS/SSL between *UA* and IdP. Among the attacks considered, the solution of [93] using HTTPS URI is only vulnerable to the phishing attack described in Sect. 6.1.2 (non-fulfillment of UBA1$_{Var1}$ and lack of an activation phase).

From this comparison results that *mID(OTP)* for *Var2* is almost comparable with HTTPS URI solution in terms of security. When a MFA is required, *mID(OTP) Var2* can also guarantee AA and UBA1$_{Var1}$, thus offering a better solution for mitigating the phishing attack.

mID(OTP) *Var1* **vs OAuth for Native Apps (HTTPS URI)**

There are three main design differences between *mID(OTP)* for *Var1* (depicted in Figure 5.2) and the abstract model recommended by the OAuth working group (depicted in Figure 2.7): the choice of the OAuth flow (Im-

Table 5.4: Comparison among OAuth Solutions and *mID(OTP) Var2*.

Asm	***mID(OTP)*** *Var2* (Fig. 5.2)	**Custom URI** (Fig. 2.7)	**HTTPS URI** (Fig. 2.7)
CA1$_{Var2}$	✓	×	✓
CA2$_{Var2}$	✓	✓	✓
CA3$_{Var2}$	✓	✓	✓
AA	✓only with MFA	n.a	n.a
UBA1$_{Var1}$	✓only with MFA	×	×

plicit flow vs Authorization Code flow), the choice of *UA* (native app vs external browser), and the activation phase.

About the flow choice, [93] does not recommend the Implicit flow as it cannot be protected by the PKCE (there is not a communication between C and the IdP to complete the security challenge, a token is directly sent to C). However, in our solution PKCE is not required, as we mitigate token interception attacks using native inter-process communication methods.

About the use of native app as *UA*, the authors of [93] do not describe any security issues. They do not recommend the use of a dedicated native application due to the increased complexity and requirement for the user to have the app installed. Nevertheless, in some scenarios, we consider the installation of ad hoc application to manage the authentication and authorization to be an advantage because it allows for easily integrating new security mechanisms; e.g., access control and a wider range of MFA solutions

A last comment is about usability versus security. The activation phase of our solution comes with a cost — in our opinion affordable — in terms of usability. However, it is the solution that better handles the phishing attack. Indeed, also the solution in [93] is heavily exposed to the phishing attack: even if in general the user can trust the *UA* by checking the locks and the navigation bar, she can be cheated by an activity created by a malicious app with all characteristics of a real browser. By contrast, during our activation phase the user will directly interact with the *UA* without passing through a possible malicious app.

Chapter 6

Phases 3-4: Security and Usability Analysis

Given the outputs of Phase 2 — the MSC of the customized model and the informal specification of its security assumptions and goals — the next phase of our methodology (Phase 3 of Figure 4.2) is analyzing its security. The security assessment is based on the *protocol assurance level* (PAL). As described in the ISO/IEC 29128 [73], a PAL represents the level of guarantees about the security protocol. The standard defines four levels and, for each level, details which kind (informal, semi-formal and formal) of protocol specification, adversarial model, security property and self-assessment evidence should be considered performing the security assessment. In our methodology, we have considered two levels (described in Table 6.1): semi-formal (*PAL1*) when the solution deals with anonymous or personal data with an evaluated risk low; and formal (*PAL3*) when the solution deals with sensitive data or the evaluated risk is high. In a semi-formal security analysis (detailed in Sect. 6.1), the IdM designer — under the informal adversarial model provided by our methodology (described in Sect. 6.1.1) — has to informally argue whether the protocol specification (MSC output of Phase 2) satisfies or not the assumptions and goals. In Sect. 6.1.3, we will provide an example for the Facebook solution. Instead, in a formal se-

curity analysis (detailed in Sect. 6.2), starting with the outputs of Phase 2, the IdM designer has to specify the protocol flow, the adversarial model, the security assumptions and goals using a formal specification language and check for possible security property violations using ~~the~~ an automated security analysis tool. Provided that the IdM designer performs the mapping (gray box of Phase 3 of Figure 4.2) between the informal specification — which we have provided in a natural-language description (output of Phase 2) — to the selected formal language, our methodology is parametric on the formal specification language. As an example, our methodology provides this formal mapping (described in Sect. 6.2.1) in ASLan++ [32], the formal language of the AVANTSSAR project [26]. Output of Phase 3 is a security analysis report. At this point, the IdM designer has to evaluate the report: did the analysis find attacks? If the answer of the security analysis is yes, Phase 2 must be performed again. Otherwise, it means that the design has an adequate degree of security and so the designer can continue with the last phase.

The final phase consists of a usability analysis that evaluates the usability aspects of the design and optionally evaluates the user satisfaction using ad-hoc questionnaires (details in Sect. 6.3). Output of this phase is a usability analysis report. At this point, the IdM designer has to evaluate the report: did the analysis identify usability problems? If the answer is yes, the IdM designer has to come back to Phase 1 and re-discuss the selected usability aspects. Otherwise the final IdM solution is obtained.

In the following sections, we will detail the security analysis (semi-formal and formal) of Phase 3 and the usability analysis of Phase 4.

6.1 Semi-formal Analysis

Given in input the protocol flow and the security properties defined in Phase 2, to validate its security, we have to informally define the adversarial model and check for possible attacks.

6.1.1 Adversarial Model

The adversarial model depends on the IdM solution taken into consideration. We have considered the behavior of a Dolev-Yao intruder [44], who can overhear and modify messages using his initial knowledge and the knowledge obtained from the traffic, and two additional attackers (or a combination of them) specific for the mobile context:

Table 6.1: Protocol Assurance Levels (from [73]).

	PAL1	**PAL3**
Protocol Specification	Semi-formal description (e.g., a MSC)	Formal description in a tool-specific specification language, whose semantics is mathematically defined
Adversarial Model	Informal description	Formal description in a tool-specific specification language, whose semantics is mathematically defined
Security Property	Informal description	Formal description in a tool-specific specification language, whose semantics is mathematically defined
Self-assessment Evidence	Informal argument that the specification of the protocol in its adversarial model achieves and satisfies its objectives and properties	Tool-aided bounded verification that the specification of the protocol in its adversarial model achieves and satisfies its objectives and properties

Malicious App: an application installed on the *User*'s device. Whether the malicious app playing the role of SP_{app}, we refer to it as *malicious* SP_{app}. Compared to a general malicious app, a malicious SP_{app} can also access security relevant parameters used in the IdM flow.

Malicious User: a human person playing the role of *User* in the IdM flow, who can extract information from her smartphone (e.g., having root permissions) or use tools to reverse engineering applications.

6.1.2 Self-assessment Evidence

Given the protocol flow, its security assumptions, goals and the adversarial model, to complete the security analysis we have to argue that the design has an adequate degree of security.

In our security analyses, given that the reference model described in Sect. 5.2.1 is based on OAuth and OpenID Connect, we took into account the security considerations reported in the OAuth specification [64, §10], where for each possible attack the respective countermeasures are described. Starting from this list of known attacks for IdM solutions, we

Table 6.2: Security Countermeasure (from [66]).

Reference	**Description**
§5.1.4.1	Enforce credential storage protection best practices
§5.1.5	Limit the token scope and the expiration time
§5.2.1	Automatic revocation of derived tokens if abuse is detected
§5.2.3.4	*AS* must issue separate client identifiers and secret to the different installations of a particular *C*
§5.3.3	Store secrets in secure storage
§5.1.4.2.2	When creating secrets *AS* should include a reasonable level of entropy
§5.1.3, §5.2.3.2	*User* should always be in control of the authorization processes and get the necessary information to make informed decision

have identified which attacks are exploitable in the case of native mobile applications. In addition, as in our work we focus on the security analysis of the design phase, we will not consider threats related to cryptographic algorithms and implementation aspects (e.g., input validation, SQL injection and DoS attacks). We assume that the common security countermeasures described in standard guidelines (e.g., in the "Digital Identity Guideline" of NIST [8] and the ENISA "Smartphone Secure Development Guidelines" [18]) are implemented. In Table 6.2, we report some security countermeasures extracted by the OAuth security guideline [66].

In our analysis, we focus on these classes of attacks:

- *Phishing*: malicious apps can create a fake login form and steal user credentials.
- *User Impersonation*: as a consequence of this attack, an attacker can login in a benign app as another user.
- *Client Impersonation*: a malicious app can impersonate another app to the *IdP* obtaining user identities and access protected resources.

6.1.3 Facebook Example

Our goal was to check whether the FB native SSO solution described in Sect. 4.2.2 is vulnerable to the attacks reported in [64].

Adversarial Model. We have considered the attackers described before: Dolev-Yao, a malicious app, a malicious *User* and their combinations.

Self-assessment Evidence. To test whether the attacks could be reproduced in the actual deployment, we used the implementation described in [10], which requires the use of the FB Android SDK. As a result of these tests,

we have discovered that, even if some implementation choices protect the FB native SSO solution from some known attacks, it anyway suffers from the following problems.

Phishing. In FB native SSO solution, as we have described in Sect. 4.2.2, the communication between C and *FB_client* is performed using explicit intents. The use of explicit intents (instead of implicit intents) mitigates phishing attacks when other malicious apps play the role of UA. In this case, the receiver of the authentication request is specified and, under the assumptions of the integrity of the OS, this ensures that only *FB_client* is invoked. However, phishing attacks are still possible in case the malicious app plays the C role. If this is the case, as pointed out in [78], malicious C can create a fake login form and steal the user credentials. In particular, when *User* clicks on the "Login with Facebook" button, malicious C shows a fake login form, identical to the one shown by *FB_client*, and it is thus able to steal the victim's credentials. This is possible as the FB solution design starts the authentication process from the C apps. Even assuming that *User* is a security expert, she has no way to discover the ongoing attack as the login form shown by a malicious C can be identical to the login form shown in the FB login flow.

Client Impersonation. A malicious C can impersonate another C with IdP_S obtaining user identities and access protected resources. As explained in Sect. 4.2.2, the FB solution is a combination of two OAuth flows, thus we have to consider two different clients: the mobile app (C) in the Implicit flow, and the *FB_client* in the RO Password Credentials flow.

In the former case, we have observed that, by performing the verification of the *key_hash* value (after Step 7 in Figure 4.5), the FB solution mitigates the client impersonation attack. Indeed, a malicious C app cannot have the same certificate of a benign C, as the certificate depends on the private

key of the C app developer.

In the latter case (when *FB_client* plays the role of C), we discovered that the client impersonation attack is exploitable. This is possible as the knowledge of the user's credentials is the only factor used by *FB_server* to authenticate *FB_client*. Thus, once obtained the user's credentials (e.g., by exploiting the phishing attack mentioned above) a malicious C can also impersonate *FB_client* sending the same message of Step 4 in Figure 4.5. To test this, we used the Advanced Rest Client,[1] allowing us to test custom HTTP requests. In detail, we have simulated the call of Step 4 of Figure 4.5 and, after entering the user's credentials, we were able to obtain a valid token. To perform this call, an attacker has to generate the *sig* parameter. In our test, we were able to discover the method used to generate the hash value by reverse-engineering the *FB_client* app. It is interesting to note that even if the *FB_client* impersonation attack is a simple consequence of a serious security issue, namely a credential leakage, it is in practice difficult to mitigate. Indeed, the issues related to credential leakage can be mitigated by making *User* aware of a possible attack. For instance, usually *IdPs* send a confirmation email to *Users* every time an access is performed through a different/new device. On the contrary, in case of the *FB_client* impersonation attack, the attacker can bypass this security measure, by obtaining a token with no expiration date and capable of accessing all the user data in a transparent way.

User Impersonation. An attacker can login in a benign C as another *User*. To reproduce this vulnerability, we have performed the attack described in Figure 6.1, where an attacker plays the role of C in Session 1, and the role of *User* in Session 2. In detail:

[1] Available at `https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddffdnphfgcellkdfbfbjeloo`.

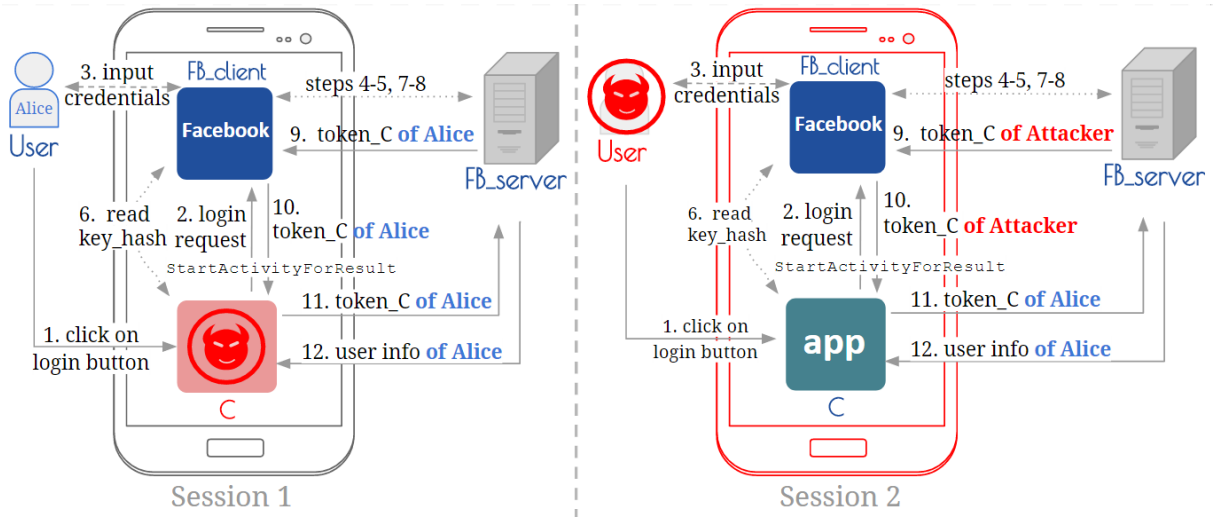


Figure 6.1: User Impersonation Attack in the FB Native SSO Solution.

- *Session 1:* a malicious C, which is installed in Alice's smartphone, obtains a valid token linked to Alice's identity (*token_C* of Alice in Step 10 of Figure 6.1) that will be maliciously used by the attacker in Session 2. To avoid that *User* becomes aware of the attack, C can finish the flow by performing Steps 11 and 12.

- *Session 2:* after the execution of the FB Login process, a benign C obtains a token linked to the attacker's identity (*token_C* of the attacker in Step 10 of Figure 6.1). Then, when the benign C exchanges this token to obtain the user's information (Step 11 of Figure 6.1), the attacker switches the token linked to his identity with the victim's token. This is possible as the attacker is the owner of the smartphone where the FB Login flow is running, and can, for example, use a proxy tool to overwrite the access token of Step 11 of Figure 6.1 with the access token previously obtained.

To mitigate user impersonation, after receiving the message of Step 10 in Figure 6.1, a C app has to check that the token (or in general the identity assertion) used to retrieve the *User* information has been granted to itself.

In the FB native SSO solution, the default token is used as a bearer token, thus it does not contain any information about the authentication process, the only way to validate the token is to request the token information to *FB_server*. The problem is that, as before, the attacker can manipulate this request.

We responsibly reported these issues to FB and, after a discussion with the FB security experts, we discovered that they provide explicit recommendations to developers on how to confirm identity: "When token is received, it needs to be verified. You should make an API call to an inspection endpoint that will indicate who the token was generated for and by which app" (for more details see [51]). Unfortunately, this mitigation works only when the *C* app has a server-side counterpart. In addition, the main login guide of FB does not make any mention of that. This mitigation is described in a page related to manually building a login flow (guideline to use when a developer implements browser-based login for an app without using FB SDKs), while this attack is possible even when the FB SDK is used.

It is interesting to note that a similar mitigation is also recommended by the OAuth working group in the OAuth extension "OAuth 2.0 Token Introspection" [68], where a method for determining meta-information about an opaque token is described.

6.2 Formal Analysis

The use of formal languages and automatic tools for analyzing security protocols has allowed researchers to uncover a large number of vulnerabilities in protocols that had been thought to be, or even informally proved to be, secure. Famous examples range from simple protocols such as the Needham-Schroeder Public Key protocol to Kerberos or TLS (see [41] for

details). These examples underline how the design of a protocol that requires specific security properties is not a simple task, as its security depends on several assumptions on trust and communication channels (e.g., the federation between the involved parties, and the transport protocol used in the message exchange). Several formal tools have been developed, such as ProVerif [34], AVISPA [27], and AVANTSSAR [26], all sharing the idea to extract from the protocol message flow a description of the entities involved, the exchanged messages and the channel assumptions. Formal protocol specifications are then given in input to automated tools that check the desired security properties of the protocol against realistic threat models.

In our formal analysis, we use ASLan++ [32], the input specification language of the AVANTSSAR Platform [26]. ASLan++ is a high-level formal language that formalizes the interactions between the different protocol roles (role-based language), where a role represents a set of operations (e.g., sending and receiving messages) that must be executed by the agent that plays that role. ASLan++ supports the specification of different channel assumptions and security goals, most notably different variants of authentication and confidentiality. After the specification in ASLan++, the model is converted in ASLan, a low-level language that can be given in input to a model checker to check if the specified security properties are satisfied. In our analysis, we use SATMC (SAT Model Checker) [29], which is one of the model checkers of the AVANTSSAR platform. SATMC can determine whether the concurrent execution of a finite number of sessions of the specified protocol satisfies the expected security properties in spite of the interference of a malicious intruder. The verification of the security properties is performed using the state-of-the-art SAT solvers and is based on the use of the linear temporal logic (as detailed in [28]).

In Sect. 6.2.1, we will present the formal specification of our reference

model.

6.2.1 Formal Specification of *mID(OTP)*

In this section, we describe how the semi-formal description of the *mID(OTP)* solution can be translated into a formal model (in this case, specified in ASLan++), detailing the initial state and the behavior of the entities, the channels and the security goals. We also describe how we have formalized the assumptions presented in Sect. 5.2.2. In Table 6.3, we show each assumption and the corresponding formal specification. In addition, we model what in Sect. 5.2.3 is indicated as *an assumption not valid or not implemented properly* by removing it from the formal model, as shown in the last column of Table 6.3.

Initial State and Behavior of Entities

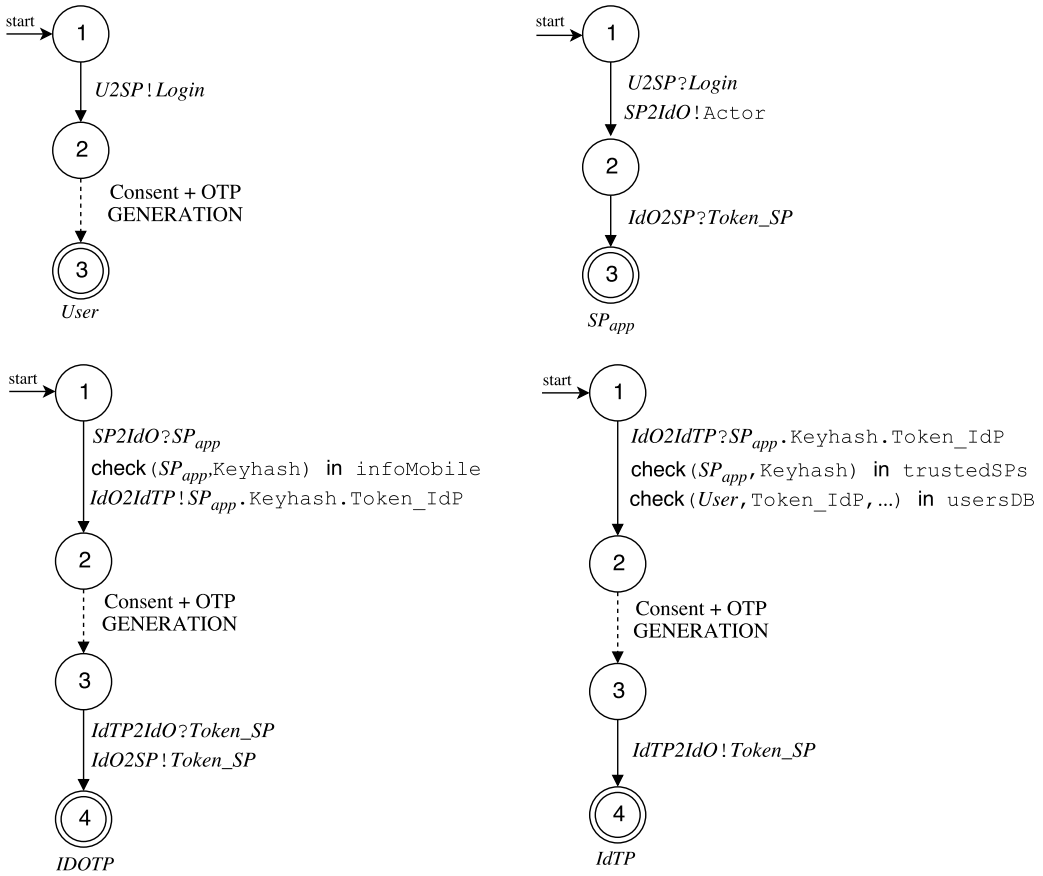
The initial state of a protocol defines the initial knowledge of the intruder, who is indicated with the letter `i`, and of all the honest entities that participate in the protocol session, where a protocol session is a particular run of the protocol, played by specific entities, using specific instances of the communication channels and optionally, additional parameters that must be passed as initial knowledge to the different entities. To model the TA assumption, as shown in Table 6.3, in our analysis we have not considered sessions with `i` playing the role of *IdTP*.

Regarding the registration phase, we have modeled the data provided by the SP_{app} developer as initial knowledge of *IdTP*. In general, after the registration phase, *IdTP* creates a database containing the relation between the SP_{app} identities, their *key_hash* values for *Var1* or *redirect_uri* for *Var2*, and the information (e.g., name and logo) provided by the SP developers. While, we have modeled the data obtained as result of the

activation phase as initial knowledge of *User*, *IDOTP* and *IdTP* for *Var1* and *OTPApp*, *Browser* and *IdTP* for *Var2*.

Table 6.3: Mapping between Assumptions (Asm(s) for short) and Formal Specification.

Asm	**Formal Specification**	
	Specification of Asm	**Removal of Asm**
TA	We do not consider sessions with `i` playing the role of *IdTP*	add sessions with `i` playing the role of *IdTP*
CA1$_{Var1}$	`link(`*IdO2IdTP*`,`*IdTP2IdO*`);`	delete `link(`*IdO2IdTP*`,`*IdTP2IdO*`);`
CA2$_{Var1}$	`authentic_on(`*SP2IdO*`,`SP_{app}`);` and DB Keyhash	delete `authentic_on(`*SP2IdO*`,`SP_{app}`);`
CA3$_{Var1}$	`confidential_to(`*IdO2IdTP*`,`*IdTP*`);`	delete `confidential_to(`*IdO2IdTP*`,`*IdTP*`);`
	`weakly_authentic(`*IdO2IdTP*`);`	`weakly_authentic(`*IdO2IdTP*`);`
	`weakly_confidential(`*IdTP2IdO*`);`	`weakly_confidential(`*IdTP2IdO*`);`
	`authentic_on(`*IdTP2IdO*`,`*IdTP*`);`	`authentic_on(`*IdTP2IdO*`,`*IdTP*`);`
	`link(`*IdO2IdTP*`,`*IdTP2IdO*`);`	`link(`*IdO2IdTP*`,`*IdTP2IdO*`);`
CA1$_{Var2}$	`confidential_to(`*B2SP*`,`SP_{app}`);`	delete `confidential_to(`*B2SP*`,`SP_{app}`);`
CA2$_{Var2}$	`confidential_to(`*B2IdTP*`,`*IdTP*`);`	delete `confidential_to(`*B2IdTP*`,`*IdTP*`);`
	`weakly_authentic(`*B2IdTP*`);`	`weakly_authentic(`*B2IdTP*`);`
	`weakly_confidential(`*IdTP2B*`);`	`weakly_confidential(`*IdTP2B*`);`
	`authentic_on(`*IdTP2B*`,`*IdTP*`);`	`authentic_on(`*IdTP2B*`,`*IdTP*`);`
	`link(`*B2IdTP*`,`*IdTP2B*`);`	`link(`*B2IdTP*`,`*IdTP2B*`);`
CA3$_{Var2}$	`confidential_to(`*O2B*`,`*Browser*`);`	delete `confidential_to(`*O2B*`,`*Browser*`);`
AA	Data obtained during the activation phase are `nonpublic` values	add all the `iknows(`$IFactor$`)` obtained during the activation phase
BA1	"Built-in": `i` cannot read the internal state of the other entities	add all the `iknows(`$IFactor_p$`)`
BA2	"Built-in": `i` cannot read the internal state of the other entities	add all the `iknows(`$IFactor_k$`)`
UBA1$_{Var1}$	`confidential_to(`*U2IdO*`,`*IDOTP*`);`	delete `confidential_to(`*U2IdO*`,`*IDOTP*`);`
UBA2$_{Var1}$	`authentic_on(`*U2IdO*`,`*User*`);`	delete `authentic_on(`*U2IdO*`,`*User*`);`
UBA1$_{Var2}$	`confidential_to(`*U2B*`,`*Browser*`);`	delete `confidential_to(`*U2B*`,`*Browser*`);`
	`confidential_to(`*U2O*`,`*OTPApp*`);`	`confidential_to(`*U2O*`,`*OTPApp*`);`
UBA2$_{Var2}$	`authentic_on(`*U2B*`,`*User*`);`	delete `authentic_on(`*U2B*`,`*User*`);`
	`authentic_on(`*U2O*`,`*User*`);`	`authentic_on(`*U2O*`,`*User*`);`

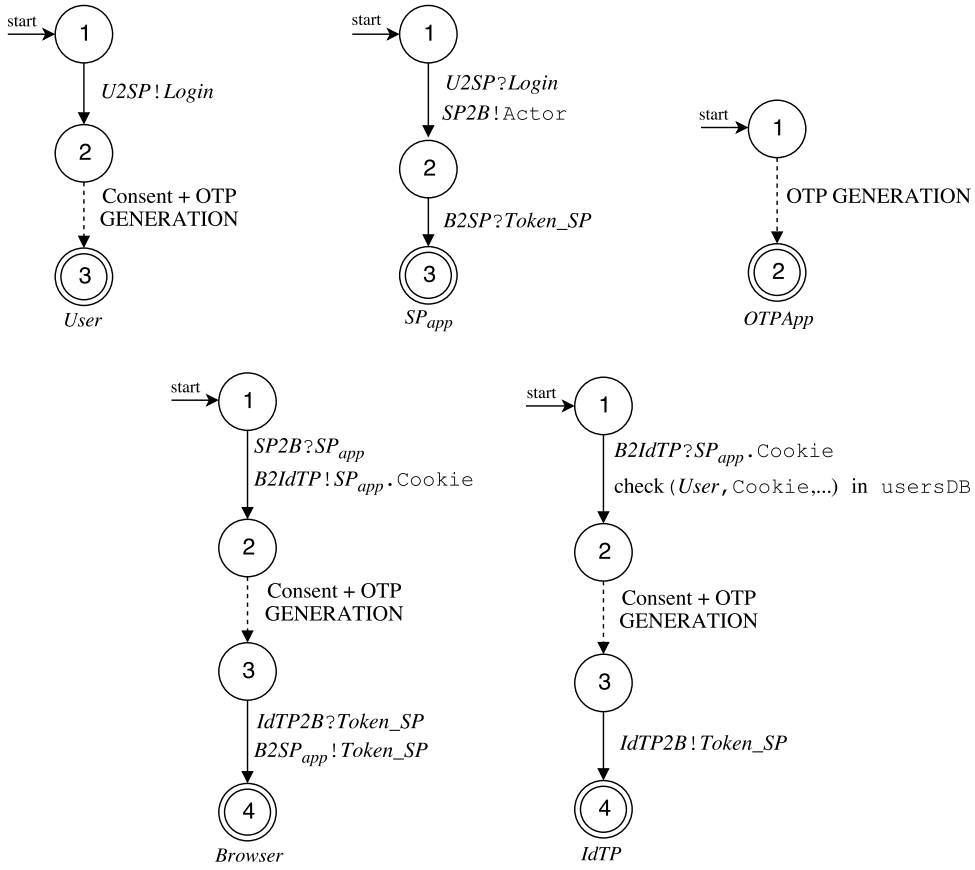


Legend:

- *U, IdO* stands for *User, IDOTP* respectively, and U2*SP, IdO2SP, IdO2IdTP, SP2IdO, IdTP2IdO* are their unidirectional channels.
- *Ch*!*M* means that message *M* is sent over channel *Ch*.
- *Ch*?*M* means that a message, says *M*, is received over channel *Ch* and a variable *X* is set to *M*.
- *M1*.*M2* is the concatenation of messages *M1* and *M2*.
- *check(X,Y,...,Z)* in *DB* means that *(X,Y,...,Z)* must be in *DB*, otherwise the protocol stops.

Figure 6.2: Protocol View for *Var1*.

To specify that the intruder knows a message m, we use the ASLan++ predicate `iknows(`m`)`. As shown in Table 6.3 for AA, BA1 and BA2, the removal of an assumption (which we will do to consider different scenarios of the analysis) boils down to adding some `iknows` facts to the initial knowledge of the intruder.



Legend:

- *U, B* stands for *User, Browser* respectively, and U2*SP, B2SP, B2IdTP, SP2B, IdTP2B* are their unidirectional channels.
- *Ch*!*M* means that message *M* is sent over channel *Ch*.
- *Ch*?*M* means that a message, says *M*, is received over channel *Ch* and a variable *X* is set to *M*.
- *M1*.*M2* is the concatenation of messages *M1* and *M2*.
- *check(X,Y,...,Z)* in *DB* means that *(X,Y,...,Z)* must be in *DB*, otherwise the protocol stops.

Figure 6.3: Protocol View for *Var2*.

The behavior of the honest entities is specified by the evolution of the system, which consists of a sequence of operations performed by each role. For simplicity, Figure 6.2 and Figure 6.3 show the evolution of the protocol using a process view, which describes the messages exchanged in Figure 5.2 and Figure 5.3 for each entity as a set of actions (e.g., receive or send a

message and DB access), where `Actor` is the keyword used in ASLan++ to represent the entity taken into consideration. These representations are not completed, as we have left the OTP generation approach and the consent steps not specified. Two complete examples are presented in Chapter 7. In Sect. 7.1, we analyze a *mID(OTP) Var1* solution using a TOTP algorithm, while in Sect. 7.3, we analyze a *mID(OTP) Var2* solution using a CR algorithm. This formal representation can be translated into various role-based formal languages and input to different state-of-the-art security protocol analyzers. In our analysis, we use ASLan++ and SATMC (see [32] for more details on language and tool).

In our analysis, we have considered the behavior of a Dolev-Yao intruder [44], who can overhear and modify messages using his initial knowledge and the knowledge obtained from the traffic — this behavior is built-in in the SATMC tool. An operation that is not allowed to `i` is the reading of the internal state of another entity, where an internal state is a list of expressions known by the corresponding entity. Thus, as highlighted in Table 6.3, BA1 and BA2 are built-in in the tool.

Formal Specification of Channels

For a detailed definition of the properties of channels between two protocol entities A and B we point the reader to [30, 87]. In a nutshell, consider a message M sent on a channel *A2B* from A to B. *A2B* is *authentic* if B can rely on the fact that only A could have sent M. *A2B* is *confidential* if A can rely on the fact that only B can receive M. *A2B* is *weakly authentic* if the channel input is exclusively accessible to a single, but yet unknown, sender, and *A2B* is *weakly confidential* if the channel output is exclusively accessible to a single, yet unknown, receiver. A *link* between two channels *A2B* and *B2A* means that the entity sending messages over the *A2B* is the same entity that receives messages from *B2A*. These properties can

be graphically represented as follows: $A \bullet\!\!\rightarrow B$, $A \circ\!\!\rightarrow B$, $A \rightarrow\!\!\bullet\ B$, $A \rightarrow\!\!\circ\ B$ meaning authentic, weak authentic, confidential and weak confidential channel, respectively; moreover, we indicate a link property between two channels with the same trace for the corresponding arrows.

As shown in Table 6.3, we have modeled as channel properties the communication assumptions (CA1_{Var1}, CA2_{Var1}, CA3_{Var1} for *Var1* and CA1_{Var2}, CA2_{Var2}, CA3_{Var2} for *Var2*) and the two user behavior assumptions (UBA1_{Var1}, UBA2_{Var1} for *Var1* and UBA1_{Var2}, UBA2_{Var2} for *Var2*). The modeling of these assumption is far from a trivial mapping and requires an explanation.

About the communication assumptions, we have that CA1_{Var1} is related to the inter-app communication in the mobile. The property expected by the `StartActivityForResult` method can be modeled by a link property between the two channels used in the mobile: the app that has sent a request is the same app that will receive the result. The property expected by the redirection scheme specified in CA1_{Var2} can be modeled with a confidential channel between *Browser* and SP_{app}. Regarding CA2_{Var1}, we have modeled an Android method, which extracts the *key_hash* value included in the package of an app, using an authentic channel (used by SP_{app} to send its identity to *IDOTP*) and a DB containing the relations between the SP_{app} identities and their *key_hash*, used by *IDOTP* to read the correct *key_hash* value. This is due to the fact that this method — executed by the Android OS — guarantees the authenticity of its output. CA3_{Var1} and CA2_{Var2} are modeled with five channel properties (see Table 6.3) that all together model a TLS/SSL unilateral channel. Finally, we have modeled CA3_{Var2}, which represents the launching of a browser after the OTP generation, as a confidential channel between TP_{app} and *Browser*.

About the user behavior assumptions, we have modeled UBA1_{Var1} and UBA2_{Var1} as properties of the channel from *User* to *IDOTP* (*U2IdO*).

UBA1$_{Var1}$ is necessary to prevent leakage of the *PIN* — entered in a malicious app or watched by an intruder during the typing — thus, we have modeled *U2IdO* as a confidential channel. UBA2$_{Var1}$ guarantees the possession of the *IDOTP* app installed in the user's smartphone. Having this assumption, only the valid *User* can communicate with that particular installation of *IDOTP*, thus we have modeled *U2IdO* as an authentic channel. Similarly with UBA1$_{Var2}$ and UBA2$_{Var2}$ considering the channels between *User* and *Browser* (*U2B*) and between *User* and *OTPApp* (*U2O*).

Formal Specification of Security Goals

For the definition of authentication we refer to [82]: whenever the entity B completes a run of the protocol apparently with the entity A, then A has previously been running the protocol apparently with B, and the two entities agree on a message M. In ASLan++, this corresponds to specifying the goal:

(G1$_{BA}$) `SP_authn_U_on_Request(_)` *User* `*->>` SP_{app};

where `*->>` indicates authenticity, directedness (i.e. the only (honest) receiver of a message is the intended one [32]) and freshness. G1$_{BA}$ requires that a message is transmitted in an authenticated and fresh manner, thus allowing SP_{app} to authenticate *User* and offering replay-protection at the same time. In addition, following the definition in [82], associated goal labels are used to specify which values of M the goal is referring to, namely, the *Login* value in State 1 of the *User* process (in Figure 6.2 and Figure 6.3) and the corresponding value in the last state of the SP_{app} process (State 3 in Figure 6.2 and Figure 6.3).

If *mID(OTP)* is used as a MFA solution, we have to consider G1$_{MFA}$ and G2. As described in Sect. 5.2.3, we have defined G1$_{MFA}$ in terms of the basic authentication goal G1$_{BA}$ and the strong and weak assumptions. This

means that, in the formal model, we consider $G1_{BA}$ and we check whether it holds under the strong assumptions and different (sub)sets of weak-assumptions. The goal must hold if the intruder is not able to compromise all the instance-factors.

Similarly, the OTP properties are checked by means of the goal:

(G2) `IDP_authn_TP_on_OTP:(_)` TP_{app} `*->>` $IdTP$`;`

where TP_{app} is played by *IDOTP* for *Var1* and by *OTPApp* for *Var2*. When the OTP generator approach requires the interaction with an external *HWToken*, this goal should be formulated for *HWToken* instead of TP_{app}.

6.3 Phase 4: Usability Analysis

The term *usability* is defined in ISO 9241-210 [74] as the "extent to which a system, product or service can be used by users to achieve goals with effectiveness, efficiency and satisfaction in a specified context of use".

We focus on usability as it is strictly related to security. There is usually a tradeoff between security and usability: more complex a system is designed to provide a better protection (e.g., multi-factor authentication) less usable it results for the final user. However, as presented in [101], sometimes improving security by reducing usability can have the consequence of making security less effective (e.g., the use of the same password to access different services). For this reason, the highest levels of security can be obtained only with an equally high level of usability. Thus, output of our methodology should be a solution that guarantees an high level of security while maximizing usability. To achieve that, our methodology considers human-centred design approach that has the goal of making usable and useful solutions, having in mind — as the center of the design process —

user's needs and requirements. In [74], a human-centred design approach is defined as an "approach to systems design and development that aims to make interactive systems more usable by focusing on the use of the system and applying human factors/ergonomics and usability knowledge and techniques".

Following our methodology, an IdM designer deals with usability aspects at two levels:

- by identifying different design choices that enable more user-friendly authentication. For example, simplifying a MFA solution by sending an OTP generated by the mobile app in a transparent way, the user does not need to remember it and enter it in a login form; or supporting a SSO experience that permits to store the user session for future interactions and for different services.

- by customizing a usability questionnaire (described in Sect. 6.3.1) and involving testers to evaluate the usability of the implemented solutions.

6.3.1 Usability Questionnaires

There are many usability questionnaires in the literature that differ in the number of questions and type of answers (e.g., open answer or rating scales). The authors of [106] give an overview of tools used to evaluate the user experience. We report the description of the most widespread usability questionnaires:

SUMI (Software Usability Measurement Inventory) was developed in the Human Factors Research Group of University College Cork in Ireland. SUMI is composed of 50 questions that deal with five usability aspects: efficiency, affect, helpfulness, control, learnability with asso-

ciated a three-point scale of agreement ("Agree", "Disagree" or "Don't Know"). Available at `http://sumi.uxp.ie/`.

WAMMI (Website Analysis and MeasureMent Inventory) was developed, as SUMI, in the Human Factors Research Group of University College Cork in Ireland. It is an online questionnaire that evaluates the user-satisfaction by asking user to compare their expectations with what they actually experience on the website. It is composed of 20 statements (some positive and some negative) with associated a five-point scale of agreement (from "Strongly Agree" to "Strongly Disagree"). The results are divided in five areas: attractiveness, controllability, efficiency, helpfulness, learnability, plus an overall usability score. Available at `http://www.wammi.com/`.

QUIS (Questionnaire for User Interaction Satisfaction) was developed by a human-computer interaction laboratory of the University of Maryland. QUIS 7.0 is composed of a demographic questionnaire, a measure of overall system satisfaction along a six scales, measures of nine specific interface factors (screen factors, terminology and system feedback, learning factors, system capabilities, technical manuals, online tutorials, multimedia, teleconferencing, and software installation) with associated nine-point scales (e.g., from "terrible" to "wonderful", from "hard" to "easy" and so one). Available at `http://www.cs.umd.edu/hcil/quis/`.

SEQ (Single Ease Question) is a single question asking users to rate the difficulty of a task on a seven-point scale (from 1 "very difficult" to 7 "very easy").

ASQ (After Scenario Questionnaire) was developed by Jim Lewis [81]. ASQ must be given after the user completes a set of tasks or a scenario,

and it is composed of the following three statement:

1. "I am satisfied with the easy of completing the tasks in this scenario." (effectiveness)
2. "I am satisfied with the amount of time it took to complete the tasks in this scenario" (efficiency)
3. "I am satisfied with the support information (online help, messages, documentation) when completing the tasks." (satisfaction)

with associated a seven-point scale of agreement (from "Strongly Disagree" to "Strongly Agree").

Our methodology provides a questionnaire template based on ASQ. We will describe it in Sect. 7.1.4 applied to a real-world scenario.

Chapter 7

Real-World Scenarios

To show the effectiveness of the methodology that we have introduced in Chapter 4 — with the corresponding reference model of Sect. 5.2 — we will apply it to four real-world scenarios: mobile electronic health (project called *TreC* - Sect. 7.1), smart city (project called *Smart Community* - Sect. 7.2), national electronic identity (project called *DigiMat-Lab* - Sect. 7.3), and digital single market (EIT activity called *FIDES* - Sect. 7.4). While for FIDES we cannot share details on the design and security assessment, for TreC, Smart Community and DigiMat-Lab we will provide the complete overview of the methodology phases.

7.1 Trec - mHealth

TreC (acronym for Cartella Clinica del Cittadino, i.e. Citizens' Clinical Record) is a platform developed in the Trentino region (Italy) for managing personal health records.[1] Besides the web platform, which is routinely used by around 80,000 users, TreC is currently designing and implementing a number of native Android applications to support self-management and remote monitoring of chronic conditions. These applications are used in a "living lab" by voluntary chronic patients according to their hospital physi-

[1]More information is available at `https://trec.trentinosalute.net/`.

cians. Examples are "TreC-Lab: Diario Diabete" that is a mobile diary that allows patients to record health data, such as the blood glucose level and physical activity; and "TreC: Referti" that permits patients to consult their personal health data and medical prescriptions from the smartphone. While in the traditional web scenario, patients access services using their local health care system credentials (leveraging a SAML-based SSO [92] solution), a solution for native SSO was currently missing. The solution we have proposed will allow patients to access different TreC e-health mobile applications (and possibly other third-party e-health applications) through a single authentication act. An implementation of the proposed model is currently tested by a set of beta testers of TreC.

In the following, we will detail the different phases of our methodology. In Sect. 7.1.1, we will identify *AppCtx* for the TreC scenario. Then, we will describe the design of the solution (Sect. 7.1.2) and we will provide a formal security analysis of it (Sect. 7.1.3). Finally, in Sect. 7.1.4, we will highlight some of the usability features that we have considered during the design and present a questionnaire to evaluate user satisfaction.

7.1.1 Phase 1: *AppCtx* Identification

The *AppCtx* of TreC is given by the values of Figure 7.1. As shown in the first row, we have instantiated the IdM roles described in Sect. 2.1 with the entities involved in the TreC scenario as follow: *User* is played by a *Patient* who wants to access her Personal Health Record (PHR) on her smartphone. IdP_S and TP_S are played by *ADC* ("Autenticazione Del Cittadino"), the local health care system. *UA* and TP_{app} are played by *OTP-PAT* managing the generation of OTPs and the SSO experience for the apps installed on the phone that are part of the federation. SP_{app} is played by $TreC_C$ that is one of the apps that are part of the *ADC* federation and it is used by *Patient* to read her PHR. Finally, SP_S is played by $TreC_S$

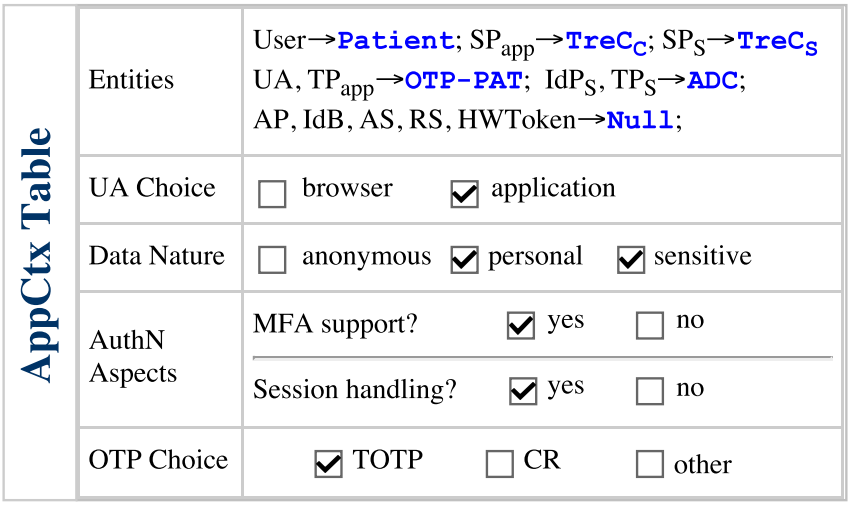


Figure 7.1: TreC *AppCtx* Table.

that manages user health data on server-side.

The *TreC* app permits patients to monitor their chronic conditions, access their PHRs, and share this information with their physicians. As it handles personal data (e.g., fiscal code, name, surname of the patients) and sensitive data (e.g., PHRs), this scenario has be designed to deal directly with security and privacy issues and the use of MFA is required. Indeed, when dealing with sensitive data, classical authentication solutions based on username-password pairs are not enough. The "European Data Protection Directive" [48] mandates that specific security measures must be implemented, including *multi-factor authentication*, a strong(er) authentication solution that, as described in Sect. 2.1.1, combines two or more authentication elements of different categories (e.g., a password combined with a code sent to a mobile device, or some biometric data). As we are considering an Italian scenario, we can refer to Figure 3.1, where a decisional diagram on Italian IdM legal obligations is shown. TreC, being a project between Fondazione Bruno Kessler, Municipality of Trento, and Azienda Provinciale dei Servizi Sanitari (APSS - Trento local health-care system) is considered as a public service dealing with personal and sensitive

data, thus CIE, CNS or SPID is required. A current problem in Italy is the adoption of SPID for mobile applications. The decree and its technical guidelines do not currently specify how to apply the SPID solution in the mobile. Thus, instead of using SPID *IdPs*, we have proposed a solution based on an *IdP* of Trento (called *ADC* - Autenticazione Del Cittadino).

This solution uses as *UA* a mobile application released by ADC, called *OTP-PAT*, which will be used as TOTP generator combined with a basic authentication with *ADC* credentials. This choice was based on the technical resources available at *ADC* (a first version of *OTP-PAT* was already on the market and used as OTP generator for accessing web applications).

Together with MFA support, this solution handles sessions between different applications. While the authentication session is valid, users can directly access all the apps in the federation, without having to enter their credentials again and again. Thus, for TreC scenario, we present the design, the formal specification and the security analysis of a solution that allows patients to access different mHealth apps through a MFA solution providing a SSO experience.

7.1.2 Phase 2: Customization of *mID(OTP)*

As *UA* is a mobile application, *OTP-PAT* corresponds to *IDOTP* of the reference model *mID(OTP) Var1* in Sect. 5.2.1. In the following sections, we will describe the registration, activation and exploitation phases (with the corresponding MSC) for the TreC scenario, and we customize the corresponding assumptions and goals.

Registration and Activation Phases of TreC

The registration phase of *TreC* is performed by the *TreC* developer following the *ADC* registration procedure. As explained in Sect. 5.2.1 for

mID(OTP) Var1, the app package name and the *key_hash* value of the *TreC* app are required metadata.

The activation phase of *TreC* is performed by *Patient* partially on her laptop and partially on her smartphone. On her laptop, *Patient* logins into the *ADC* web portal using her CNS (the card is read by desktop smartcard reader), and generates a temporary code (it lasts 5 minutes). On her smartphone, *Patient* downloads the *OTP-PAT* application using an official marketplace. Then, she digits the temporary code together with her *ADC* credentials into *OTP-PAT*. If the login is successful, the activation phase is completed by *Patient* with a creation of a *PIN* code. As a consequence of the activation, *OTP-PAT* obtains from *ADC* a *token_IdP* and a seed value that is stored encrypted with the *PIN* code ($\{|seed|\}\_PIN$) selected by *Patient*.

Exploitation Phase of TreC and MSC

Figure 7.2 shows the A-steps of the exploitation phase of *mID(OTP)* for this scenario. Compared to Figure 5.2, we have detailed the user consent and OTP generation box (Step A5), and graphically shown the channel properties as explained in Sect. 6.2.1. Given that $TreC_S$ is not involved in the A-steps, for the sake of brevity, in the rest of the section we refer to $TreC_C$ simply with *TreC*. Steps A6 (a-d) model the behavior of a Time-OTP (TOTP) algorithm [63], which is a time synchronization algorithm that generates OTPs as a function of the time of the execution and a seed (i.e. a shared secret). In general, a TOTP algorithm requires that "the prover and verifier must either share the same secret or the knowledge of a secret transformation to generate a shared secret" [63], without specifying when and how to exchange this secret. In the analyzed use-case, *OTP-PAT* obtains the seed value as part of the activation phase, and then stores it encrypted with the PIN code ($\{|seed|\}\_PIN$) selected by the *Patient*. Thus,

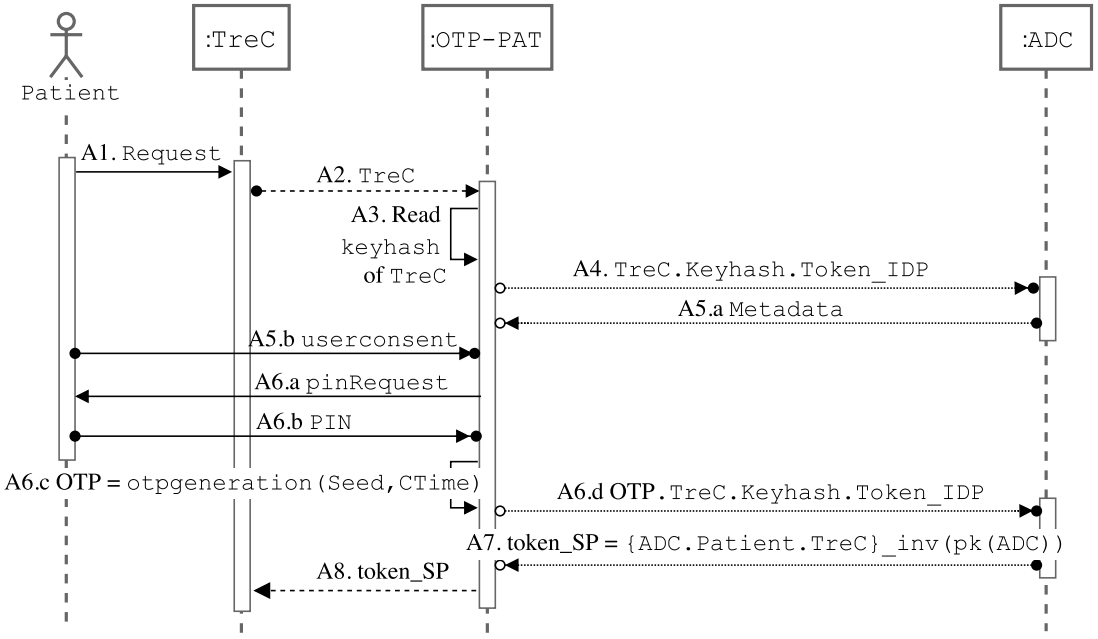


Figure 7.2: MSC of the Exploitation Phase of TreC.

the OTP generation box depicted in Figure 5.2 is replaced here with a PIN request (Step A6.a), the entering of the PIN (Step A6.b), the generation of the OTP as a function of the seed — extracted using the PIN as decryption key — and of time (Step A6.c), and finally, the request to *ADC* with the OTP value (Step A6.d).

This scenario corresponds to a MFA with 3 instance-factors: $token\_IdP$, $\{|seed|\}\_PIN$ as $IFactor_o$, and PIN as $IFactor_k$.

Assumptions and Goals of TreC

Table 7.1 shows the security assumptions and goals for the TreC scenario.

7.1.3 Phase 3: Security Analysis

As *TreC* deals with sensitive data, our methodology requires a formal security analysis. First, we have formally specified (using ASLan++) the protocol flow of Figure 7.2 and instantiated the security assumptions and

goals of *mID(OTP) Var1* for this scenario. The formalized model, assumptions and goals are then given as input to SATMC. The results of our security analysis are presented at the end of this section.

Table 7.1: Asms and Goals for TreC.

Asm	**Informal Description**
TA	*ADC* is trusted by *TreC* on identity assertions.
$CA1_{Var1}$	The communication between *TreC* and *OTP-PAT* is carried over an inter-app communication implemented using `StartActivityForResult()`.
$CA2_{Var1}$	To read the *key_hash* value (Step A3 of Figure 7.2), *OTP-PAT* uses the Android method `getPackageInfo(client packageName, PackageManager.GET SIGNATURES)`, which extracts the information about the certificate fingerprint included in the package of *TreC*.
$CA3_{Var1}$	The communication between *OTP-PAT* and *ADC* occurs over a unilateral SSL or TLS channel (henceforth SSL/TLS), established through the exchange of a valid certificate (from *ADC* to *OTP-PAT*).
AA	The activation phase is correctly performed by *Patient*. That is, *Patient* downloads the correct *OTP-PAT* (it is not fake app) and correctly follows the process, and the communication channels used are secure.
BA1	Integrity and confidentiality of data stored in the device, i.e. an app cannot read or modify data stored by another app.
BA2	There is no surveillance software (e.g., keylogger) installed on the user's device capable of reading the values that *Patient* types.
$UBA1_{Var1}$	*Patient* enters her credentials and PIN only in the correct *OTP-PAT* app being careful not to be seen by other people
$UBA2_{Var1}$	*Patient* is the only person using *OTP-PAT* that is associated to her identity.
Goals	**Informal Description**
$G1_{MFA}$	Goal ($G1_{BA}$) (i.e. *TreC* authenticates *Patient*) holds even if an intruder knows up to 2 instance-factors.
G2	The OTP must prove its origin (*ADC* authenticates *OTP-PAT*, as it is the only app knowing a secret value shared with *ADC*), and it is non-reusable (*ADC* accepts only one OTP for a specific operation avoiding replay attacks).

Initial States.

To model the TA assumption, as shown in Table 7.2, in our analysis we have not considered sessions with `i` playing the role of *ADC*.

Regarding the registration phase, we have modeled the data provided by the *TreC* developer as initial knowledge of *ADC*. As shown in Table 7.2 by the AA assumption, we have modeled the data obtained as result of the activation phase (*token_IdP* and data required for generating OTPs) as initial knowledge of *User*, *OTP-PAT* and *ADC*. In particular, as result of the activation phase: a *Patient* knows her PIN value (`pinUser`), *OTP-PAT* knows `token_IDP` and `{|seed|}_pinUser`, and *ADC* creates a DB (`usersDB`) with `Patient`, `token_IDP` and `seed` as entry.

Behavior of Entities.

Figure 7.3 shows the evolution of the protocol using a process view, which describes the messages exchanged in Figure 7.2 for each entity as a set of actions. The translation of the process view into ASLan++ is quite straightforward. The complete ASLan++ specification can be found in Appendix B.1. Here, we provide only an example by considering Steps 1 and 2 of Figure 7.3, which involve the entities *Patient*, *TreC* and *OTP-PAT*. Focusing on *TreC*, this exchange of messages in ASLan++ corresponds to

```
Patient -Ch_P2T-> Actor: Request; %Step 1
Actor -Ch_T2O-> OTPPAT: Actor;   %Step 2
```

where `Actor` is the keyword used in ASLan++ to represent the entity taken into consideration, in our example *TreC*.

Formal Specification of Security Goals

The security goals of Sect. 6.2.1 are instantiated in the following way:

(G1$_{BA}$) `SP_authn_U_on_Request:(_) Patient *->> TreC;`

with the associated goal labels specifying for M the `Request` value in State 1 of the *Patient* process (in Figure 7.3) and the corresponding value in the last state of the *TreC* process (State 3 in Figure 7.3).

Similarly, the OTP properties are checked by means of the goal:

(G2) `IDP_authn_TP_on_OTP:(_) OTPPAT *->> ADC;`

with the associated goal labels specifying M the values `otp_generation(Seed, Time)` in State 3 of both the *OTP-PAT* and the *ADC* processes in Fig-

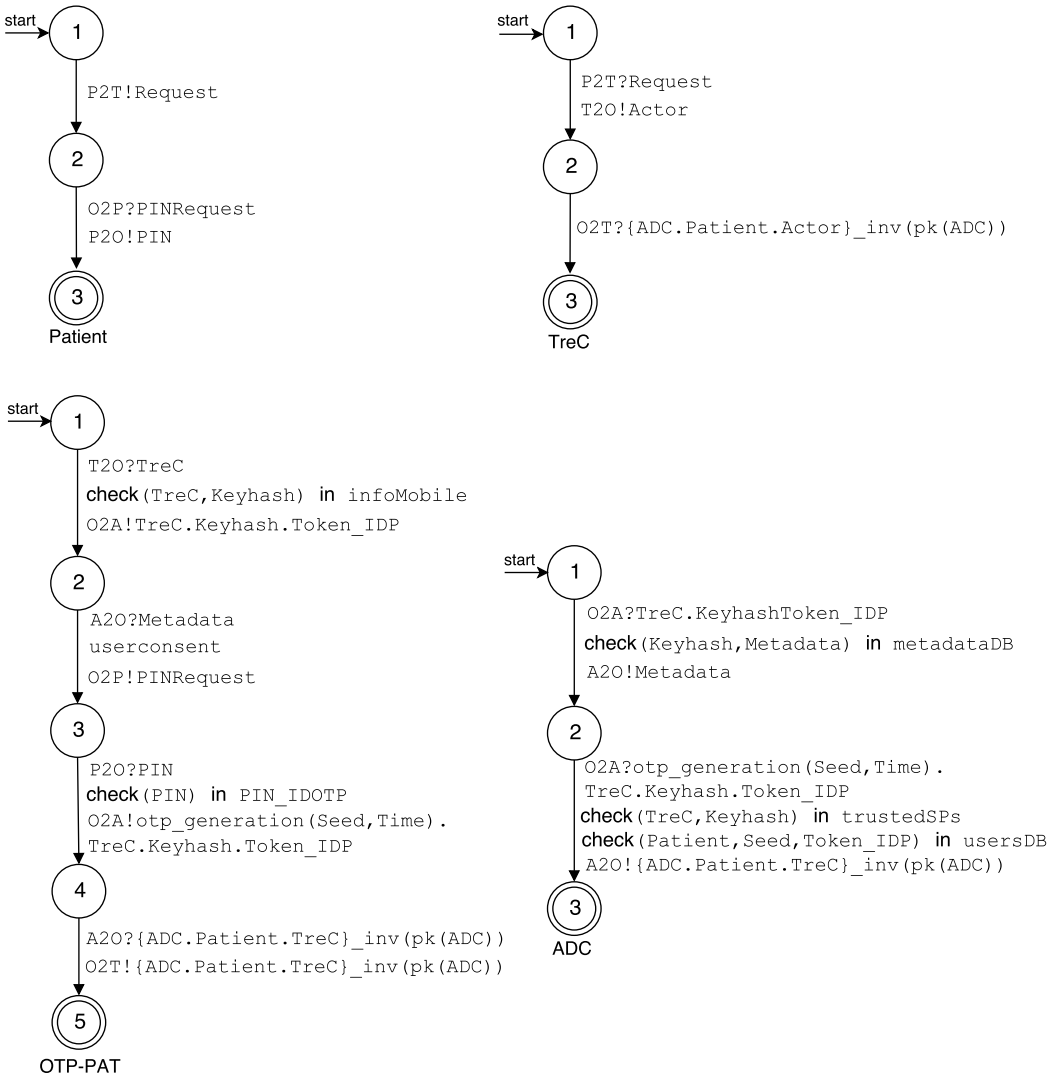
Table 7.2: Mapping between Asm(s) and Formal Specification for TreC.

Asm	**Formal Specification**	
	Specification of Asm	**Removal of Asm**
TA	We do not consider sessions with `i` playing the role of *ADC*	add sessions with `i` playing the role of *ADC*
CA1$_{Var1}$	`link(T2O,O2T);`	delete `link(T2O,O2T);`
CA2$_{Var1}$	`authentic_on(T2O,TreC);` and DB Keyhash	delete `authentic_on(T2O,TreC);`

CA3$_{Var1}$	`confidential_to(O2A, ADC);`
	`weakly_authentic(O2A);`
	`weakly_confidential(A2O);`
	`authentic_on(A2O,ADC);`
	`link(O2A,A2O);`
		delete `confidential_to(O2A,ADC);`
		`weakly_authentic(O2A);`
		`weakly_confidential(A2O);`
		`authentic_on(A2O,ADC);`
		`link(O2A,A2O);`

AA	Data obtained during the activation phase are `nonpublic` values	add `iknows(pinUser);`
		`iknows(token_IDP);`
		`iknows({|seed|}_pinUser);`

BA1	"Built-in": `i` cannot read the internal state of the other entities	add `iknows(token_IDP);` `iknows({|seed|}_pinUser);`
BA2	"Built-in": `i` cannot read the internal state of the other entities	add `iknows(pinUser);`
UBA1$_{Var1}$	`confidential_to(P2O,OTPPAT);`	delete `confidential_to(P2O,OTPPAT);`
UBA2$_{Var1}$	`authentic_on(P2O,Patient);`	delete `authentic_on(P2O,Patient);`



Legend:

- P, T, O, and *A* stands for Patient, TreC, OTP-PAT, and ADC respectively, and P2T, P2O, O2P, O2T, O2A, T2O, A2O are their unidirectional channels.
- *Ch*!*M* means that message *M* is sent over channel *Ch*.
- *Ch*?*M* means that a message, says *M*, is received over channel *Ch* and a variable *X* is set to *M*.
- *M1*.*M2* is the concatenation of messages *M1* and *M2*.
- *check(X,Y,...,Z)* in *DB* means that *(X,Y,...,Z)* must be in *DB*, otherwise the protocol stops.
- *M*_inv(pk(ADC)) means that message *M* is digitally signed with the private key of ADC.

Figure 7.3: Protocol View of TreC Exploitation Phase.

ure 7.3. Where we have modeled `Seed` as a constant value shared between *OTP-PAT* and *ADC*, and `Time` as a session parameter (cf. [32]) shared between *OTP-PAT* and *ADC*. Thus, *ADC* will accept only one OTP value

for each session, enforcing the non-reusable property.

Results of the Security Analysis

We are now ready to discuss the results of the security assessment that we have performed on the TreC scenario. Our focus is determining whether the concurrent execution of a finite number of protocol sessions enjoys the expected security goals in spite of the intruder. To this aim, we have mechanically analyzed the formal model of our scenario using SATMC, a state-of-the-art model checker for security protocols. SATMC carries out an iterative deepening strategy on k. Initially k is set to 0, and then it is incremented till an attack is found (if any) or k_{max} is reached. If this is the case, no attack traces of length up to k_{max} exist. The trace includes the actions performed by attacker and honest participants. The length of the trace is counted taking into account the parallel execution of not conflicting actions (actions executed in parallel are considered as a single step). Most of the actions of the attacker are executed in parallel (and counted as a single step) with the ones of honest participants. We set k_{max} to 1.5 times the length of the longest trace of the protocol when only honest entities participate. As a rule of thumb, with this choice we are reasonably confident that no attack is possible with greater values of k_{max}. In our analysis, the length of the longest trace of the protocol when only honest entities participate is 19, thus we have set $k_{max} = 30$. We have considered several cases including (at most) three parallel sessions in which the intruder either does not play any role or plays the role of SP_{app} (the *TreC* app in the scenario). In each session, we used different instances of the channels. The complete set of specifications can be found in Appendix B.1.

In Sect. 5.2.2, in relation to the security goal $\mathrm{G1}_{MFA}$ (and consequently to $\mathrm{G1}_{BA}$), we have described a list of strong and weak assumptions that

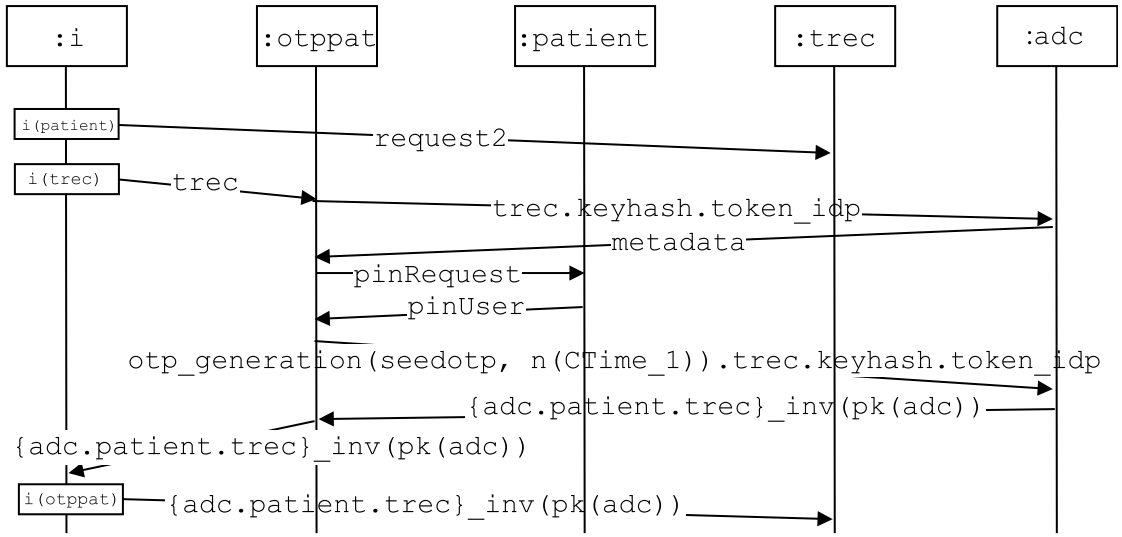


Figure 7.4: Attack Trace without the Strong Assumption CA2_{Var1}.

we have added to the model to constrain the intruder's abilities. Table 7.3 summarizes the security analyses that we have performed to check this goal.

Regarding the strong assumptions (TA, CA1_{Var1}, CA2_{Var1}, CA3_{Var1} and AA), we have performed the following analyses:

Analysis 1: We have checked that by removing only one of the five strong assumptions from the model we have a violation of G1_{BA} (i.e. there is an attack). For this analysis, we have thus performed 5 executions of SATMC removing one strong assumption at a time. To provide an example of an attack, Figure 7.4 shows the attack trace deriving from removing CA2_{Var1}. In this attack, `i` can impersonate `trec` simply be-

Table 7.3: Analyses performed for G1_{BA} - *TreC* Scenario.

Analysis	**Strong Asm(s)**	**Weak Asm(s)**	**Atk**
1	all -1	all	Yes
2	all	all -1	No
3	all	all $-m$ $(1 < m \leq 4)$	*

cause the channel used to exchange its identity is not authentic; thus, `i` can pretend to be another app. Note that, for the sake of clarity, this figure (and the other figures shown in this section) represents only the significant steps of the attack traces found by the SATMC tool.

Regarding the weak assumptions (BA1, BA2, UBA1_{Var1}, and UBA2_{Var1}), we have performed the following analyses that are detailed in Table 7.4:

Analysis 2: We have checked that by removing only one of the four weak assumptions from the model, SATMC does not find any attack on the solution (i.e. the intruder is not able to impersonate the user). Indeed, as shown in Table 7.4, by removing only one weak assumption, the intruder obtains only 1 or 2 instance-factors.

Analysis 3: We have checked that by removing specific subsets of weak assumptions it is possible to compromise all the instance-factors, causing a violation of G1_{BA}. In Table 7.3, the star (*) denotes that the result can be "yes" or "no" depending on the chosen subset of weak assumptions. The subsets shown in Table 7.4 violate G1_{BA} and result in different attack traces. Figure 7.5 shows the attack trace deriving from removing UBA1_{Var1} and UBA2_{Var1} (e.g., a proximity intruder that watches the *PIN* entered by *Patient* and then steals the smart-

Table 7.4: Results for G1_{BA} (Analyses 2 and 3).

Removed Weak Asm(s)	**Compromised Factors**			**Atk**
	PIN	**{seed}_PIN**	**token_IdP**	
BA1	×	✓	✓	No
BA2	✓	×	×	No
UBA1_{Var1}	✓	×	×	No
UBA2_{Var1}	×	✓	✓	No
$(\text{UBA1}_{Var1} \vee \text{BA2}) \wedge \text{BA1}$	✓	✓	✓	Yes
$(\text{UBA1}_{Var1} \vee \text{BA2}) \wedge \text{UBA2}_{Var1}$	✓	✓	✓	Yes

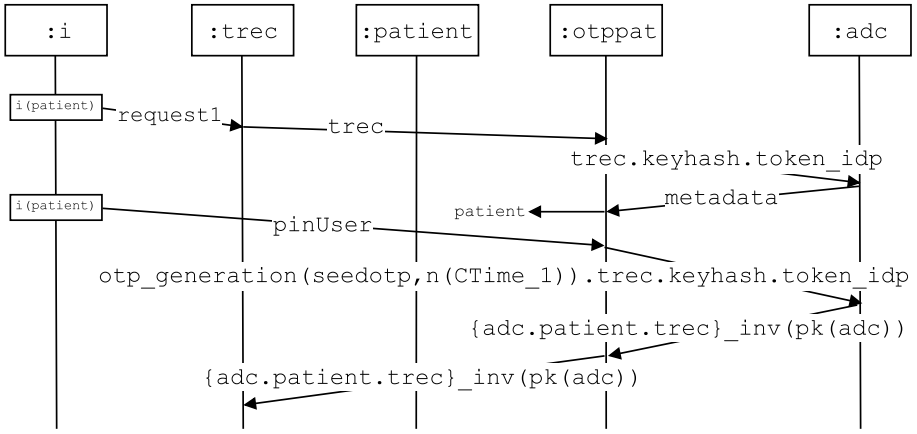


Figure 7.5: Attack Trace obtained removing $\text{UBA1}_{Var1} \wedge \text{UBA2}_{Var1}$.

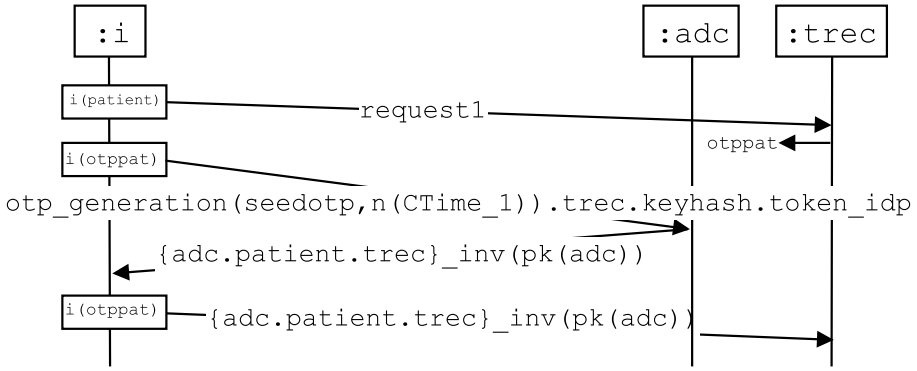


Figure 7.6: Attack Trace obtained removing BA1 ∧ BA2.

phone). In the attack, `i` initiates a session of the protocol with `trec` pretending to be `patient` (indicated as `i(patient)`). By entering the PIN code (`pinUser`) when requested by `otppat`, `i` is able to impersonate the patient and obtaining the requested resource (`resources2`). Figure 7.6 shows the attack trace deriving from removing both BA1 and BA2 (e.g., a hacker that steals the *PIN* typed by *Patient* using a keylogger and reads *token_IdP* and $\{|seed|\}\_PIN$ exploiting a malware installed on the smartphone). In this case, `i` is able to generate an OTP and sends a token request to `adc`.

As expected, when checking the solution w.r.t. the security goal G2 — which embodies the OTP properties — under all the (weak and strong)

assumptions, SATMC does not find any attack.

7.1.4 Phase 4: Usability Analysis

In the TreC usability report, we deal with usability aspects at two levels: evaluating the design choices and asking about a hundred of TreC testers to fill a questionnaire to evaluate their satisfaction.

Design Choices

In addition to the establishment of high-level security for authentication solutions for mHealth apps, it is essential to take the usability aspect into consideration. Monitoring apps often require a daily or even hourly use, but understandably users cannot be bothered by a long and complex authentication procedure each time they want to read or update their data, especially on mobile devices where the keyboard is small and sometimes uncomfortable to use. Thus, instead of asking for each access username, password and OTP, we augmented the usability in two ways:

- The designed solution does not ask *Patient* to digit the OTP; after the *PIN* input, the OTP value is sent to *ADC* in a transparent way.
- The designed solution provides a SSO experience: a session token is stored in *OTP-PAT*. Until the session is valid, *Patient* has to digit only her *PIN* to access *TreC* or other federated apps.

TreC Questionnaire

We have decided to evaluate the usability of the activation and exploitation phases using two questionnaires that are based on ASQ (cf. Sect. 6.3.1). We have split the evaluation in two questionnaires to permit users to better understand the difference between the activation and the exploitation phase. In particular, we plan to send the questionnaire on the activation

phase to users after the installation and configuration of *OTP-PAT*, and the questionnaire on the exploitation phase after two-week use of *TreC* and *OTP-PAT*.

Questionnaire on the Activation Phase. It is composed of three sections. It starts with two yes or no questions to classify the user level: *new* (if both answers are no), *some experience* (if one answer is yes and the other is no), and *expert* (if both answer is yes). Section 2 must be filled if the user succeeds in the installation and activation, and it is composed of four seven-point scale questions — from 1 "strongly agree" to 7 "strongly disagree" — about effectiveness, efficiency, and satisfaction, and an open question to leave suggestions. Finally, Section 3 must be filled if the user does not succeed in the installation and activation, and it is composed of an open question to leave some comments about the encountered difficulties.

Section 1. Please, answer with YES or NO:

1. In the last 3 months, have you access online services using a second-factor authentication method (e.g., using a token or a smartcard)?

2. Did you succeed in the *OTP-PAT* installation and activation?

Section 2. If the installation and activation of *OTP-PAT* succeeds:

1. Overall, I am satisfied with the easy of completing the activation of *OTP-PAT*.

   STRONGLY AGREE 1 2 3 4 5 6 7 STRONGLY DISAGREE

2. Overall, I am satisfied with the amount of time it took to complete the activation of *OTP-PAT*.

   STRONGLY AGREE 1 2 3 4 5 6 7 STRONGLY DISAGREE

3. Overall, I am satisfied with the support information (e.g., tutorial presentation in power-point and online documentation) when completing the activation of *OTP-PAT*.
   STRONGLY AGREE 1 2 3 4 5 6 7 STRONGLY DISAGREE

4. Overall, I am think that the activation phase is designed to guarantee a secure access to my health-data in the following exploitation phase.
   STRONGLY AGREE 1 2 3 4 5 6 7 STRONGLY DISAGREE

5. Please, leave us some comments on the activation phase (e.g., suggestions to simplify it)

   _____

   _____

   _____

Section 3. If the installation and activation of *OTP-PAT* do not succeed:

1. Which was your encountered difficulties during the installation and activation of *OTP-PAT*?

   _____

   _____

   _____

Questionnaire on the Exploitation Phase. It is composed of three sections. It starts with a yes or no question to know if the user has succeed in accessing her PHRs using *TreC* and *OTP-PAT*. If the user succeeds, Section 2 must be filled. It is composed of two seven-point scale questions — from 1 "strongly agree' to 7 "strongly disagree" — about effectiveness and efficiency, and an open question to leave suggestions. Finally, Section 3 must be filled if the user does not succeed, and it is composed of an open question to leave some comments about the encountered difficulties.
Section 1. Please, answer with YES or NO:

1. Did you succeed in accessing your PHRs using *TreC* and *OTP-PAT*?

Section 2. If you succeed:

1. Overall, I am satisfied with the easy of accessing *TreC* after the digit of a PIN in *OTP-PAT*.

   STRONGLY AGREE 1 2 3 4 5 6 7 STRONGLY DISAGREE

2. Overall, I am satisfied with the amount of time it took to access *TreC*.

   STRONGLY AGREE 1 2 3 4 5 6 7 STRONGLY DISAGREE

3. Please, leave us some comments on the exploitation phase (e.g., suggestions to simplify it)

   _____

   _____

   _____

Section 3. If you do not succeed:

1. Which was your difficulties during the access of *TreC* using *OTP-PAT*?

   _____

   _____

   _____

7.2 Smart Community - Smart Cities

In this section, we discuss the main security issues underlying the design of an access delegation mechanism for smart city mobile apps and present a solution overcoming these security problems. We start by explaining the IdM needs for a smart city scenario (Sect. 7.2.1). Then, we describe Smart Community project and the related platform (Sect. 7.2.2). Finally,

we apply our methodology to this scenario detailing the *AppCtx* identification (Sect. 7.2.3), the design that we have proposed (Sect. 7.2.4), and the security and usability analysis that we have performed (Sect. 7.2.5 and Sect. 7.2.6).

7.2.1 Smart Cities and Open Data

In recent years, city authorities are increasingly opening their back-end services through the use of Application Programming Interfaces (APIs), in particular to allow community organizations and businesses to interact with city open data. This holds the promise to foster civic engagement (through greater transparency), to create a more efficient delivery of government services, and enable a new wave of local industry innovation. There is a clear trend in considering APIs as essential building blocks of Smart Cities. To witness this trend, several projects (such as City Service Development Kit [3], Oracle's Smart City Platform Solution [15], and FIWARE [12]) offer open APIs to support the easy and rapid development of Smart City services and applications.

Despite its importance, the effective exploitation of APIs must face several challenges ranging from bureaucratic barriers — which come with enacting any widespread change — to technical impediments — such as those deriving from the fact that so much of the available data is kept in silos. Today, most applications follow the recurrent pattern of consuming data (either open data or personal user information) stored by other applications or services. For instance, a mobile app for personal mobility may wish to retrieve the bus timetables — exposed as open data by the public transportation system — and the current location of the user — retrieved by the GSM capability of the mobile device on which the app is running. APIs play a crucial role to open up data silos in a way that supports the development of smart services and applications that foster innovation, new

business opportunities, and smart capabilities for citizens (see the discussion in [55] for e-government services).

To permit the flexible combination of interface functionalities from different APIs that are typical of innovative applications, there is a need for mechanisms that break the information silos and allow applications to retrieve the data they need while maintaining security (especially those of users). A possibility is that data owners enable an application willing to access their data to authenticate as the data owners. This is clearly undesirable, since it allows the application consuming data to impersonate the data owners and inherit all their access privileges on the data. A much better solution is a mechanism by which the data owner can delegate access to a selected subset of the data to a designated client application, without enabling the client application to impersonate the data owner. The mechanism delegated access is composed of two steps: one in which client applications can request access to selected pieces of information and one in which users instruct servers to grant (or deny) requested accesses.

We have designed a delegated access in the context of the Smart Community Platform, whose development is supported by a joint project between Fondazione Bruno Kessler and Provincia Autonoma di Trento for deploying open services for Smart Cities in the Trentino region; more information available at `http://www.smartcommunitylab.it/` (cf. Sect. 7.2.2). An implementation of our solution is routinely used by around 13,000 users of the platform since 2015. By the time of writing, we have not been reported any security issue.

7.2.2 Smart Community Platform

The ambition of the Smart Community (SC) platform is to support the development of smart communities in the Trentino region by providing a set of core services (and applications) that citizens can use in their ev-

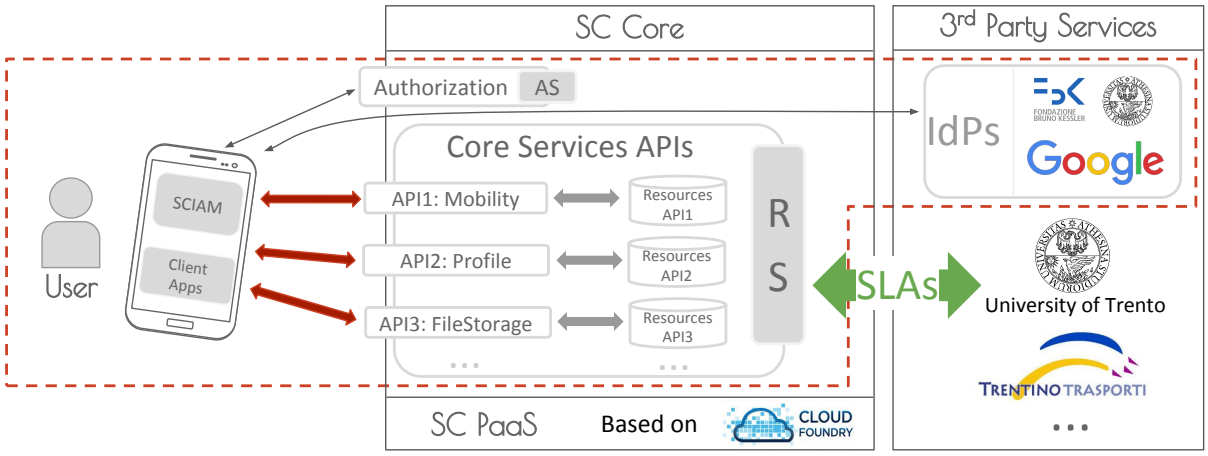


Figure 7.7: The Architecture of the Smart Community (SC) Platform.

eryday activities. The high-level architecture of the platform is depicted in Figure 7.7. *Users* is a citizen willing to exploit the *Client Apps* made available by SC or by third-party developers. The underlying SC Platform-as-a-Service (PaaS) layer, based on Cloud Foundry [4], offers an open environment in which developers can deploy server-side applications. In the following, we focus on the Software-as-Service (SaaS) layer of the SC platform which is composed of the *SC Core Services* and the *Third Party Services* offering functionalities that can be exploited by Client Apps.

The SC Core Services include components for authentication and authorization together with services providing information extracted from various resources such as SC user profiles, mobility and public transportation in the Trento area, repositories of files that are internal to the platform or offered by cloud providers (e.g., Amazon and Dropbox). All these resources can be accessed by appropriate APIs usually based on the REST paradigm. The Third Party Services are external (usually legacy) services with respect to the SC platform; e.g., services from the University of Trento for students or from the company for public transportation of the Trento area, called Trentino Trasporti. The exploitation of these external services via the SC platform is regulated by Service Level Agreements (SLAs) that

are enforced by appropriate service wrappers. Note that, also the authentication process is delegated to external IdPs, specifically University of Trento, FBK and Google. Below, we focus on the interplay among Users, Client Apps, Google IdP, and the Core Services of the SC platform, (cf. the dashed box in Figure 7.7) while abstracting away the interaction between Third Party Services and the core of the platform.

7.2.3 Phase 1: *AppCtx* Identification

The *AppCtx* of the SC scenario is given by the values of Figure 7.8. As shown in the first row, we have instantiated the IdM roles described in Sect. 2.1 with the entities involved in the SC scenario as follow: *User* is played by a *User* who wants to access her SC data on her smartphone. SP_{app} is played by C that is one of the apps that are part of the SC federation and it is used by the *User* to access her data. *UA* is played by *SCIAM* that stands for "Smart Community Identity Access Management" and is a mobile application released by SC to enable the execution of the processes of (delegated) authorization and (user) authentication, needed by a C app to access the resources from the platform, by exchanging messages with SC_AS and Google IdP_S. Finally, *AS*, *IdB* are played by SC_AS, and *RS* is played by SC_RS.

SC platform processes data of different nature. They are mainly anonymous data, such as the bus timetable. However, to manage user data, SC needs to store a personal profile of each user, thus dealing also with personal data. Referring to the decisional diagram on Italian IdM legal obligations of Figure 3.1, we are in the case of a private company dealing with personal data. As SC scenario, only anonymous or personal data are collected, the solution does not required a MFA. The basic authentication with Google is selected and a token to provide a SSO experience with session handled between apps is added.

AppCtx Table		
	Entities	User → **User** SP_{app}→**C**; UA → **SCIAM**; AS, IdB→**SC_AS**; RS→**SC_RS**; IdP_S→**Google**; SP_S, AP, TP_S, TP_{app}, HWToken →**Null**;
	UA Choice	☐ browser ☑ application
	Data Nature	☑ anonymous ☑ personal ☐ sensitive
	AuthN Aspects	MFA support? ☐ yes ☑ no Session handling? ☑ yes ☐ no
	OTP Choice	☐ TOTP ☐ CR ☐ other

Figure 7.8: SC *AppCtx* Table.

7.2.4 Phase 2: Customization of *mID(OTP)*

The SC solution can be informally described as a solution that enables *C* apps (3rd Party or SC native mobile apps) to securely access resources made available through the SC platform and permitting *Users* — after an authentication — to grant or deny access to their personal data (e.g., those in their user profiles). Therefore, the SC scenario extends *mID(OTP)* considering also authorization aspects, thus requiring some extra design steps. The starting reference model is *mID(OTP) Var1* of Sect. 5.2.1. However, compared to *IDOTP*, *SCIAM* is not released by the IdP_S (played in our scenario by Google), thus it cannot ask for Google credentials. As we will explain in the activation phase, our solution will combine the Google authentication with the exchange of a session token to provide a SSO experience inside the SC federation.

In the following sections, we will describe the registration, activation and exploitation phases (with the corresponding MSC) for the SC scenario, and we customize the corresponding assumptions and goals.

Registration and Activation Phases of SC

Goal of the registration phase of SC is to enable a C app to interact with *SCIAM* to obtain access to (selected parts of the) resources in SC_RS. Together with the exchange of the C app meta-data and the *key_hash* value (as described in Sect. 5.2.1 *Var1*), the app developer has to select which parts of the resources in the SC platform the C app needs to access for its operations from the list of available ones. The SC platform decides which access permissions to grant or deny according to some criteria (e.g., reputation of developers, sensitivity of the resources, constraints deriving from the SLAs signed with third parties). At the end of this phase, a client identifier (C_id) is returned to C.

SCIAM is not released by IdP_S (played in our scenario by Google), thus it cannot ask for Google credentials. Thus, to perform the authentication process required in the activation phase, SCIAM integrates the social login with Google that, in turn, is integrated in Android OS and is based on OpenID Connect. The activation phase of SC is performed by *User* on her smartphone. *User* downloads the *SCIAM* application using an official marketplace. Then, she enters the application and clicks on "Login with Google". After *User* login with Google, *SCIAM* obtains from Google a user token (called *google_token*) that is exchanged with SC_AS to obtain a *token_IdP*. To be sure that *token_IdP* is exchanged with a correct *SCIAM* app, SC_AS must check the *SCIAM* identity by analyzing the *google_token*. This is a JWT (JSON Web Token) [67] containing the identifier of Google, *User* and *SCIAM* app.

Exploitation Phase of SC

The exploitation phase of SC corresponds to the one described in Sect. 5.2.1 for *Var1* plus two additional steps to access user's resources (Steps 8-

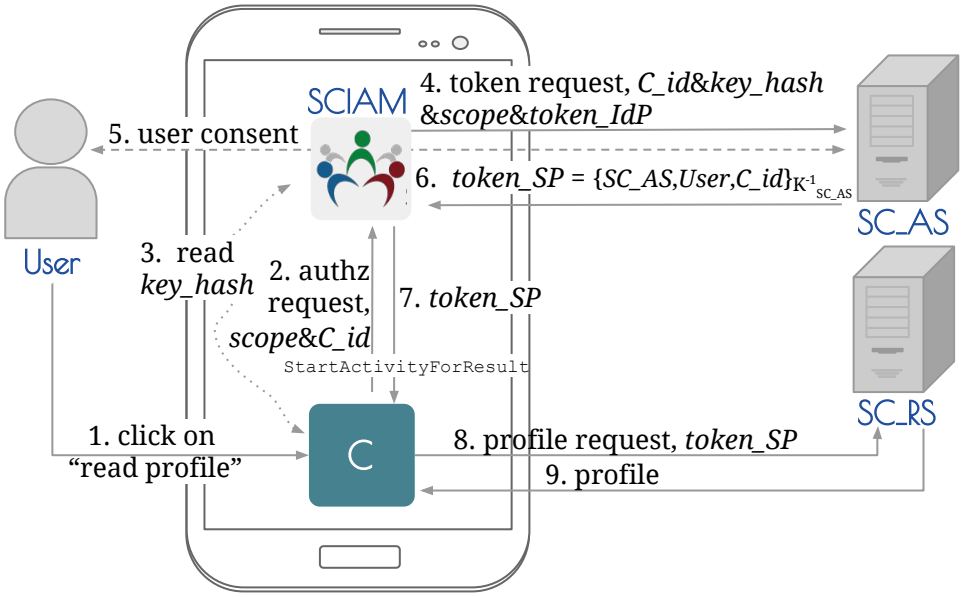


Figure 7.9: Exploitation Phase of SC.

9). The goal is to enable a C app to access the resources made available by SC_RS. Figure 7.9 shows the flow of messages of the authorization process. In Step 1, while using C, *User* clicks on a button to read her SC profile. In Step 2, C invokes *SCIAM* (by using the method `startActivityForResult`) and passes the client identifier C_id together with a parameter (called *scope*) specifying a list of permissions requested to RS. As a result, C is put on hold for *SCIAM* to return a result. In Step 3, *SCIAM* reads the certificate fingerprint (*key_hash*) of C and, in Step 4, *SCIAM* sends a request — containing C_id, *key_hash*, *scope*, and *token_IdP* — to SC_AS to get a (fresh) access token enabling C to claim the requested permissions. SC_AS checks if the provided *key_hash* matches the one previously provided in the registration phase for the given C_id, and the set of interface functionalities to be invoked on the resources contained in the *scope* parameter is a subset of the approved APIs for C. If one of the two conditions is false, the flow is interrupted and an error is shown to *User*; otherwise (Step 5) SC_AS presents *User* with a form

asking her whether to authorize C to access her SC profile. In Step 6, if *User* agrees, *SC_AS* sends a response to *SCIAM* containing *token_SP*. In Step 7, *SCIAM* returns *token_SP* to C to conclude the invocation of the method `startActivityForResult` in Step 2. If *token_SP* is valid, C can use it to retrieve the user profile from *SC_RS* (Steps 8 and 9).

Assumptions and Goals of TreC

Table 7.5 shows the security assumptions and goals for the SC scenario. Compared to the reference model, we have added a goal specific for authorization.

7.2.5 Phase 3: Security Analysis

For the SC solution, as no sensitive data are involved the corresponding protocol assurance level is PAL1. Thus, we have performed a semi-formal security analysis of our solution as the one described in Sect. 6.1.3 for Facebook solution, considering the following attacks: phishing, user impersonation and client impersonation. To mitigate phishing attacks, we have integrated our solution with Google login. In this case, *Users* do not need to enter their Google credentials in an untrusted applications as a login session is usually stored in an Android phone. Regarding user impersonation, which permits to an attacker to login in a benign C app as another *User*, we adopt JWTs (this is inspired to what is done in the OpenID Connect [94]) that contain a set of claims about the authentication session, including an identifier for *User*. Finally, our solution mitigates the client impersonation attacks in two ways. First, the risk of *SCIAM impersonation* is reduced by using *google_token* (which is different for every user-*SCIAM* as it is generated during the activation phase) for authenticating *SCIAM* with *SC_AS*. Second, *third-party client app impersonation* can be prevented by making *SC_AS* to check the certificate fingerprint

(*key_hash*) of C (recall Steps 3 and 4 of the authorization process): a malicious app cannot obtain the certificate of C as this depends on its private key.

Table 7.5: Asms and Goals for SC.

Asm	**Informal Description**
TA	*Google* is trusted by *SCIAM* on identity assertions.
$CA1_{Var1}$	The communication between C and *SCIAM* is carried over an inter-app communication implemented using `StartActivityForResult()`.
$CA2_{Var1}$	To read the *key_hash* value (Step A3 of Figure 7.2), *SCIAM* uses the Android method `getPackageInfo(client packageName, PackageManager.GET SIGNATURES)`, which extracts the information about the certificate fingerprint included in the package of C.
$CA3_{Var1}$	The communication between *SCIAM* and *SC_AS* occurs over a unilateral SSL or TLS channel (henceforth SSL/TLS), established through the exchange of a valid certificate (from *SC_AS* to *SCIAM*).
AA	The activation phase is correctly performed by *User*. That is, *User* downloads the correct *SCIAM* (it is not fake app) and correctly follows the process, and the communication channels used are secure.
BA1	Integrity and confidentiality of data stored in the device, i.e. an app cannot read or modify data stored by another app.
BA2	There is no surveillance software (e.g., keylogger) installed on the user's device capable of reading the values that *User* types.
$UBA1_{Var1}$	*User* enters her credentials in the correct *Google* form invoked by the *SCIAM* app being careful not to be seen by other people.
$UBA2_{Var1}$	*User* is the only person using *SCIAM* that is associated to her identity.
Goals	**Informal Description**
$G1_{BA}$	C authenticates *User*
G3	*User* delegates access to C for accessing her resources in *SC_RS*

7.2.6 Phase 4: Usability Analysis

For SC scenario, we have dealt with usability aspect only at design level. The exploitation phase of SC was designed having in mind a daily use. Thus, instead of asking for each access username and password, our solution improves usability by supporting a SSO experience for user authentication. In addition, to alleviate the burden of *C* app developers to integrate with SC platform and external services, our solution offers a uniform interface to manage and enforce authorizations.

7.3 DigiMat-Lab - Italian Electronic Identity

DigitMat-Lab is a joint laboratory between FBK and IPZS (acronym for "Istituto Poligrafico Zecca dello Stato"), which is the Italian state printing office and mint. We are involved in various activities, one is the design of a mobile multi-factor authentication mechanism that uses the Italian electronic identity card (CIE 3.0 - Carta d'Identità Elettronica) [85] as an OTP generator. CIE is a personal identification document that will replace the paper-based identity card in Italy and is used for both online and offline identification. The microprocessor of CIE is contactless and can be read using smartphone with a NFC (Near Field Communication) interface.

7.3.1 Phase 1: *AppCtx* Identification

The *AppCtx* of the DigiMat-Lab scenario is given by the values of Figure 7.10, where we have instantiated this authentication mechanism for a smart-city service. As shown in the first row, we have instantiated the IdM roles described in Sect. 2.1 with the entities involved in the DigiMat-Lab scenario as follow: *User* is played by a *Citizen* who wants to access *Car* — a mobile app for car-sharing — on her smartphone using *CIE*, which is

AppCtx Table		
	Entities	User→**Citizen**; SP_{app}→**Car**; SP_S→**Car_S** UA→**Browser**; TP_{app}→**IPZS**; IdP_S, TP_S→**IdP_{IPZS}**; HWToken→**CIE**; AP, IdB, AS, RS→**Null**;
	UA Choice	☑ browser ☐ application
	Data Nature	☐ anonymous ☑ personal ☐ sensitive
	AuthN Aspects	MFA support? ☑ yes ☐ no Session handling? ☑ yes ☐ no
	OTP Choice	☐ TOTP ☑ CR ☐ other

Figure 7.10: DigiMat-Lab *AppCtx* Table.

a smartcard and plays the role of *HWToken*. SP_{app} and SP_S are played by *Car* and Car_S, respectively, and are one of the apps that are part of the federation with IdP_{IPZS}, which is a SAML *IdP* that plays the role of IdP_S and TP_S. *UA* is played by *Browser* that manages the SSO experience for the apps installed on the phone that are part of the federation. Finally, TP_{app} is played by *IPZS* that manages the interaction with *CIE*.

Similarly to the TreC scenario, we present the design, the formal specification and the security analysis of a solution that allows citizens to access different mobile apps through a MFA solution providing a SSO experience.

The reason of the *UA* choice is based on the *IdP* nature. To be in line with the majority of *IdPs* of the public administration and, in perspective, with SPID, we need a solution that supports SAML, which is a browser-based authentication protocol. For security and usability reasons, as described in Sect. 4.1.2, we select an external in-app browser.

The choice of the OTP approach is linked to the CIE nature. This card carries a X.509 certificate issued by the Certification Authority of Ministry of Internal Affairs with its public and private keys. Encryption, decryption and signature with this certificate are the basic functionalities of this card, that is therefore suitable for a CR OTP approach (cf. Sect. 2.1.1).

7.3.2 Phase 2: Customization of *mID(OTP)*

This scenario is based on *mID(OTP) Var2*, where *Browser* plays the *UA* role and the OTP is generated using a CR approach. In the following sections, we will describe the registration, activation and exploitation phases (with the corresponding MSC) for the DigiMat-Lab scenario, and we customize the corresponding assumptions and goals.

Registration and Activation Phases of DigiMat-Lab

The registration phase of DigiMat-Lab is performed by the *Car* developer following the *IPZS* registration procedure. As explained in Sect. 5.2.1 for *mID(OTP) Var2*, the app package name and the *redirect_uri* value of the *Car* app are required metadata.

The activation phase of DigiMat-Lab is performed by *Citizen* on her smartphone using *CIE*. First, *Citizen* downloads *IPZS*. Then, *Citizen* opens *IPZS* that redirects her into *Browser* for the authentication with IdP_{IPZS}. As result of the authentication process *Browser* obtains a cookie, called *session_cookie*, that is used (from here on) to identify the user session in place of the user credentials. If the authentication succeeds, *IPZS* asks *Citizen* to digit the *PIN* code of *CIE* (a value known to *Citizen*) and to put in contact *CIE* with her smartphone. In the interaction between *CIE* and *IPZS* the *PIN* value is checked and the user X.509 certificate (*CertU*) is exchanged. At the end of the activation phase, *IPZS* stores *CertU* and the second part of the *PIN* code (called *PIN2*).

Exploitation Phase of DigiMat-Lab

Figure 7.11 shows the A-steps of the exploitation phase of *mID(OTP) Var2* for this scenario. Compared to Figure 5.3, we have detailed the OTP generation box (Steps A4 a-h), the user consent box (Steps A5 a-c),

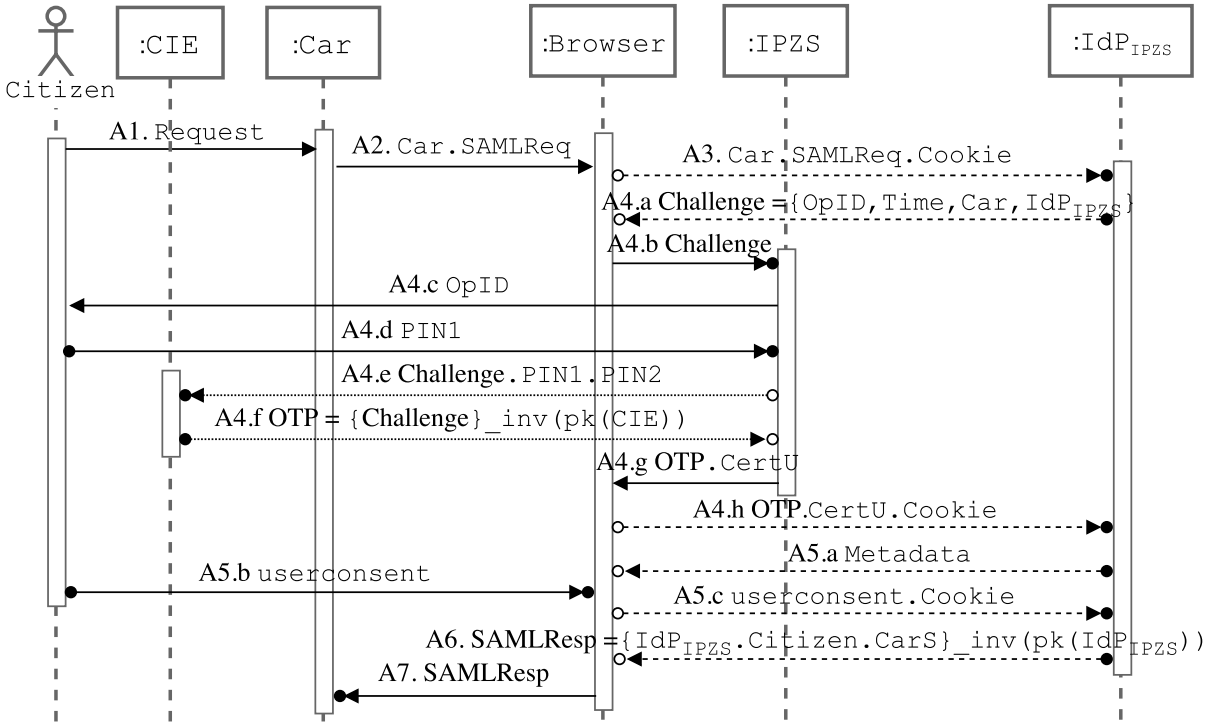


Figure 7.11: MSC of the exploitation phase of DigiMat-Lab.

and graphically shown the channel properties as explained in Sect. 6.2.1. In addition, being IdP_{IPZS} a SAML-based *IdP*, Steps A2 and A3 require an additional parameter, called *SAMLRequest*. This parameter is sent from Car_S to *Car* during the S-steps. To complete the SAML protocol — after the authentication — *Car* sends a *SAMLResponse*, which corresponds to *token_SP* of the reference model, to Car_S. Being SAML a browser-based authentication protocol, an alternative solution is to involve Car_S during the A-steps. In Step A3, *Car* can send a login request to Car_S that redirects, through the *Browser*, the *SAMLRequest* to IdP_{IPZS}. Similarly, at the end of the authentication IdP_{IPZS} can send, through the *Browser*, a *SAMLResponse* directly to Car_S.

Steps A4 a-h model the behavior of a CR algorithm. A challenge is generated by the IdP_{IPZS} (Steps A4.a) and is sent to *IPZS* (Steps A4.b). In Steps A4 c-d, *IPZS* asks *Citizen* to digit the first part of her *PIN* (called

PIN1) specifying the operation (in this case the login with *Car*). *IPZS* combines the two parts *PIN1* (value just typed) and *PIN2* (value stored during the activation phase) of the *PIN* and start the communication with *CIE* via NFC. In Step A4.e, *IPZS* sends the challenge to *CIE*. After checking the *PIN* value, *CIE* signs the challenge using the private key stored inside the chip and sends the generated OTP value to *IPZS* (Step A4.f). In Step A4.g, *IPZS* sends, through *Browser*, the OTP value and *CertU* to IdP_{IPZS}. In Step A4.h, *Browser* will add the corresponding *session_cookie*. If the authentication succeeds, IdP_{IPZS} returns to *Browser* a consent containing the meta-data of *Car* (Step A5.a). In Step A5.b, *Citizen* checks whether *Car* is the app that she wants to access and decides whether to give her consent or not. In Step A5.c, *Browser* sends the consent decision to IdP_{IPZS} together with the corresponding *session_cookie*. If *Citizen* agrees, in Step A6 IdP_{IPZS} returns a *SAMLResponse* to *Browser* that, in Step A7, redirects it to *Car* following its *redirect_uri* (value stored by IdP_{IPZS} during the *Car* registration phase).

This scenario corresponds to a MFA with 4 instance-factors: *PIN2*, *session_cookie*, *CIE* as $IFactor_o$, and *PIN1* as $IFactor_k$.

Assumptions and Goals of DigiMat-Lab

In this scenario, compared to *mID(OTP) Var2*, a *HWToken* (*CIE*) is involved. Thus, as a new entity and a new communication channel are involved, we need to consider two additional assumptions:

(CA4) The communication between *CIE* and *IPZS* occurs over a secure channel, established through the exchange of a valid certificate from *CIE* to *IPZS*.

(UBA3) *IPZS* is the only app that can communicate with *CIE*.

Note that, with UBA3 we are assuming that *CIE* is not stolen or there are not malicious mobile applications installed on the user's smartphone ables to communicate with *CIE*. Table 7.6 shows the security assumptions and goals for the DigiMat-Lab scenario.

7.3.3 Phase 3: Security Analysis

In our instantiation, *Car* does not deal with sensitive data. However, if there are some damages to the car, to ask a compensation to the user, *Car* needs to know the user identity with a high level of assurance, thus requiring a MFA solution analyzed with a formal security tool. In addition, *IPZS* can be federated with other services dealing with sensitive data. For this reason we have performed a security analysis of PAL 3 Level.

First, we have formally specified (using ASLan++) the protocol flow of Figure 7.11 and instantiated the security assumptions and goals of *mID(OTP) Var2* for this scenario. The formalized model, assumptions and goals are then given as input to SATMC. The results of our security analysis are presented at the end of this section.

Initial States.

To model the TA assumption, as shown in Table 7.2, in our analysis we have not considered sessions with `i` playing the role of IdP_{IPZS}.

As shown in Table 7.7 by the AA assumption, we have modeled the data obtained as result of the activation phase as initial knowledge of *IPZS*, *Browser* and IdP_{IPZS}. In particular, as result of the activation phase: *IPZS* knows user *PIN* value and stores the second part (`pin2`), *Browser* knows `session_cookie`, and IdP_{IPZS} creates a DB (`usersDB`) with `Citizen` and `session_cookie` as entry.

Behavior of Entities.

For computational problem, we have slightly simplify the flow of Figure 7.11 analyzing the evolution of the protocol shown in Figure 7.12.

Table 7.6: Asms and Goals for DigiMat-Lab.

Asm	**Informal Description**
TA	IdP_{IPZS} is trusted by *Car* on identity assertions.
$CA1_{Var2}$	*Browser* to exchange *SAMLResponse* with *Car* (Step A7 of Figure 7.11) uses Android App Links.
$CA2_{Var2}$	The communication between *Browser* and IdP_{IPZS} occurs over a unilateral SSL or TLS channel (henceforth SSL/TLS), established through the exchange of a valid certificate (from IdP_{IPZS} to *Browser*).
$CA3_{Var2}$	*IPZS* uses the Chrome Custom Tabs to launch *Browser*.
CA4	The communication between *CIE* and *IPZS* occurs over a secure channel, established through the exchange of a valid certificate from *CIE* to *IPZS*.
AA	The activation phase is correctly performed by *Citizen*. That is, *Citizen* downloads the correct *IPZS* (it is not fake app) and correctly follows the process, and the communication channels used are secure.
BA1	Integrity and confidentiality of data stored in the device, i.e. an app cannot read or modify data stored by another app.
BA2	There is no surveillance software (e.g., keylogger) installed on the user's device capable of reading the values that *Citizen* types.
$UBA1_{Var2}$	*Citizen* enters her credentials only in a trusted *Browser*, and the PIN value for the OTP generation only in the correct *IPZS* app, being careful not to be seen by other people.
$UBA2_{Var2}$	*Citizen* is the only person using *Browser* and *IPZS* that are associated to her identity.
UBA3	*IPZS* is the only app that can communicate with *CIE*.
Goals	**Informal Description**
$G1_{MFA}$	Goal ($G1_{BA}$) (i.e. *Car* authenticates *Citizen*) holds even if an intruder knows up to 3 instance-factors.
G2	The OTP must prove its origin (IdP_{IPZS} authenticates *CIE*, as it is the only app knowing user private key), and it is non-reusable (IdP_{IPZS} accepts only one OTP for a specific operation avoiding replay attacks).

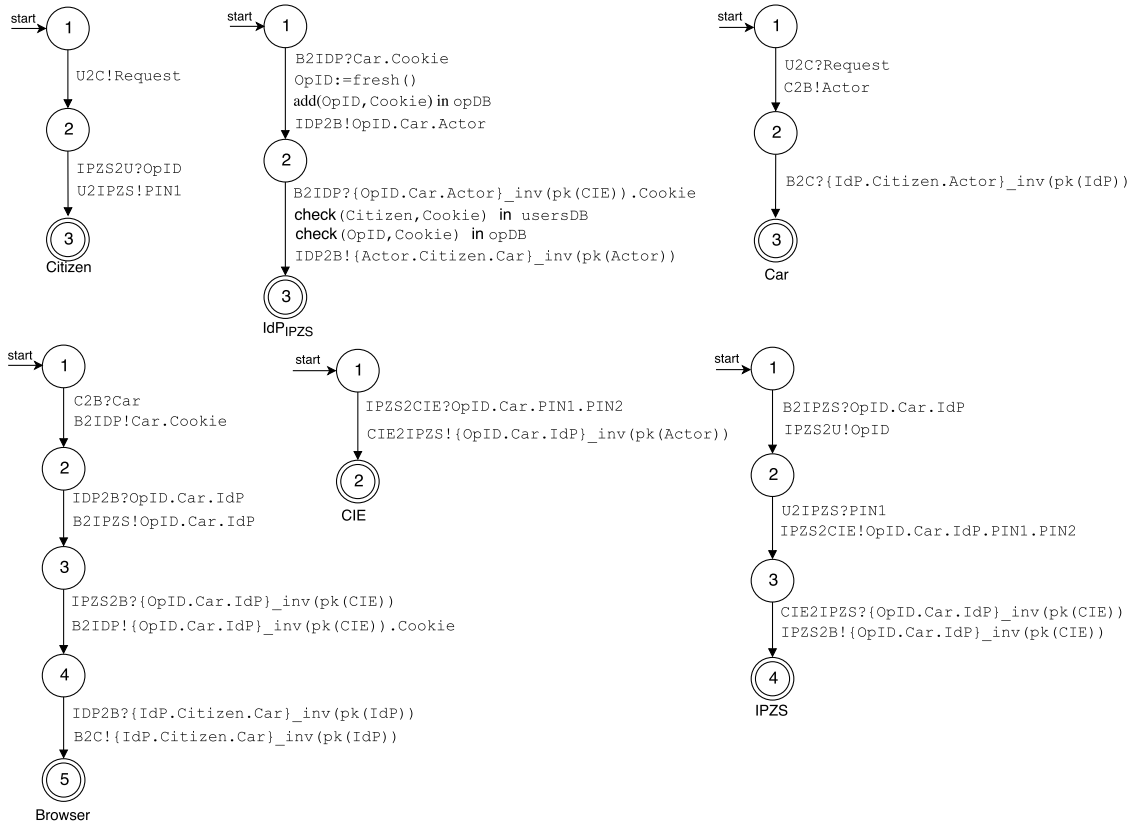
The complete ASLan++ specification can be found in Appendix B.2.

Formal Specification of Channels

Table 7.7 shows the channels assumptions ($CA1_{Var2}$, $CA2_{Var2}$, $UBA1_{Var2}$ and $UBA2_{Var2}$) of Sect. 6.2.1 instantiated for this scenario and the formal

Table 7.7: Mapping between Asm(s) and Formal Specification for DigiMat-Lab.

Asm	**Formal Specification**	
	Specification of Asm	**Removal of Asm**
TA	We do not consider sessions with `i` playing the role of *IdP*	add sessions with `i` playing the role of *IdP*
$CA1_{Var2}$	`confidential_to(B2C,Car);`	delete `confidential_to(B2C,Car);`
$CA2_{Var2}$	`confidential_to(B2IdP,IdP);`	delete `confidential_to(B2IdP,IdP);`
	`weakly_authentic(B2IdP);`	`weakly_authentic(B2IdP);`
	`weakly_confidential(IdP2B);`	`weakly_confidential(IdP2B);`
	`authentic_on(IdP2B,IdP);`	`authentic_on(IdP2B,IdP);`
	`link(B2IdP,IdP2B);`	`link(B2IdP,IdP2B);`
$CA3_{Var2}$	`confidential_to(IPZS2B,Browser);`	delete `confidential_to(IPZS2B,Browser);`
CA4	`confidential_to(IPZS2CIE,CIE);`	delete `confidential_to(IPZS2CIE,CIE);`
	`weakly_authentic(IPZS2CIE);`	`weakly_authentic(IPZS2CIE);`
	`weakly_confidential(CIE2IPZS);`	`weakly_confidential(CIE2IPZS);`
	`authentic_on(CIE2IPZS,CIE);`	`authentic_on(CIE2IPZS,CIE);`
	`link(IPZS2CIE,CIE2IPZS);`	`link(IPZS2CIE,CIE2IPZS);`
AA	Data obtained during the activation phase are `nonpublic` values and `i` can interact with `CIE`	add `iknows(PIN1); iknows(PIN2); iknows(session_cookie);` and delete `authentic_on(IPZS2CIE,IPZS);`
BA1	"Built-in": `i` cannot read the internal state of the other entities	add `iknows(session_cookie); iknows(PIN2);`
BA2	"Built-in": `i` cannot read the internal state of the other entities	add `iknows(PIN1);`
$UBA1_{Var2}$	`confidential_to(U2IPZS,IPZS);`	delete `confidential_to(U2IPZS,IPZS);`
$UBA2_{Var2}$	`authentic_on(U2IPZS,Citizen);`	delete `authentic_on(U2IPZS,Citizen);`
UBA3	`authentic_on(IPZS2CIE,IPZS);`	delete `authentic_on(IPZS2CIE,IPZS);`



Legend:

- U, C, B, and IdP stands for Citizen, Car, Browser, and IdP_{IPZS} respectively, and U2C, IPZS2U, U2IPZS, C2B, B2C, B2IDP, IDP2B, B2IPZS, IPZS2B, IPZS2CIE, CIE2IPZS are their unidirectional channels.
- *Ch*!*M* means that message *M* is sent over channel *Ch*.
- *Ch*?*M* means that a message, says *M*, is received over channel *Ch* and a variable *X* is set to *M*.
- *M1*.*M2* is the concatenation of messages *M1* and *M2*.
- *check(X,Y,...,Z)* in *DB* means that *(X,Y,...,Z)* must be in *DB*, otherwise the protocol stops.
- *M*_inv(pk(CIE)), *M*_inv(pk(IdP)) means that message *M* is digitally signed with the private key of CIE and IdP, respectively.

Figure 7.12: Protocol View of DigiMat-Lab Exploitation Phase.

modeling of the new channel assumptions (CA3_{Var2} and UBA3) on the channels between *CIE* and *IPZS*.

With UBA3 we are assuming that *CIE* is not stolen or there are not malicious mobile applications installed on the user's smartphone ables to communicate with *CIE*, thus we can modeled this assumption as a confidential challenge from *IPZS* (the trust app used by the user) and *CIE*.

The idea of CA3_{Var2} is that, before the challenge exchange of Step A4.e

CIE and *IPZS* establish a secure channel performing the following steps: Diffie-Hellman key exchange, external authentication (where *IPZS* authenticates with *CIE* through the exchange of a certificate) and internal authentication (where *CIE* authenticates with *IPZS* through the exchange of a valid certificate). Even if it seems like a mutual authentication, the certificate use by *IPZS* (in general the terminal that is able to read the card) is generated using some public values, thus is not a real authentication (more details are specified in [86]). This assumption can be modeled with five channel properties (see Table 7.7) that all together model this secure channel.

Formal Specification of Security Goals

The security goals of Sect. 6.2.1 are instantiated in the following way:

(G1$_{BA}$) `SP_authn_U_on_Request:(_) Citizen *->> Car`;

with the associated goal labels specifying for M the `Request` value in State 1 of the *Citizen* process (in Figure 7.12) and the corresponding value in the last state of the *Car* process (State 3 in Figure 7.12).

Similarly, the OTP properties are checked by means of the goal

(G2) `IDP_authn_TP_on_OTP:(_) CIE *->> IdP;`

with the associated goal labels specifying M the values {`OpID.Car.IdP`}_`inv(pk(CIE))` in State 2 of both the *CIE* and the corresponding value in the last state of the IdP_{IPZS} process (State 3 in Figure 7.12).

Results of the Security Analysis

We are now ready to discuss the results of the security assessment that we have performed on the DigiMat-Lab scenario. While in TreC analysis, to be on the safe side, we set $k_{max} = 30$ as max attack trace length, in this

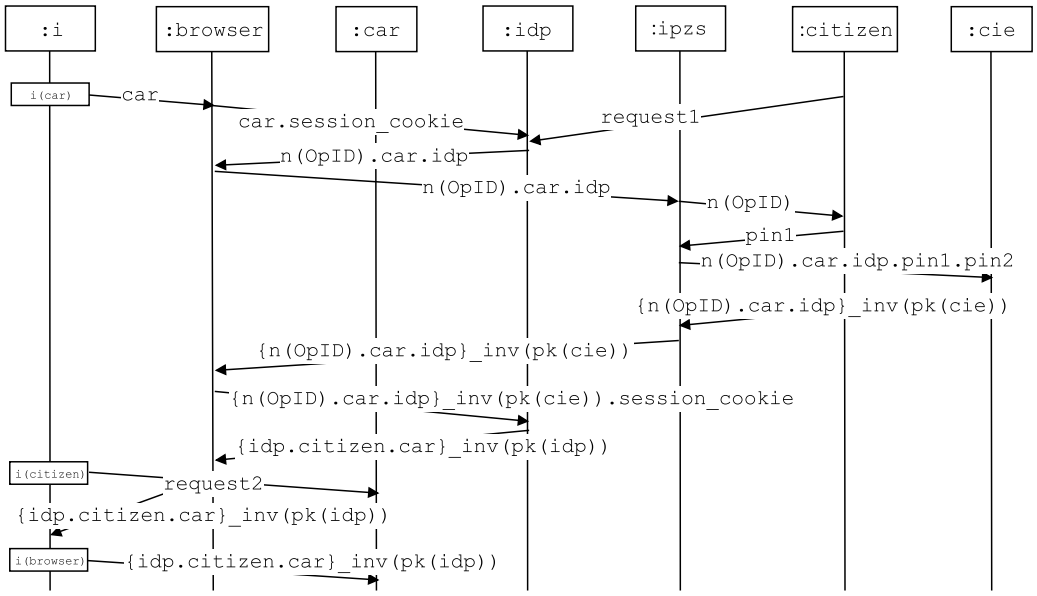


Figure 7.13: Attack trace without the strong assumption CA1$_{Var2}$.

scenario we need at least $k_{max} = 44$ (1.5 times 29, the longest trace of the protocol when only honest entities participate). We have considered several cases including (at most) three parallel sessions in which the intruder either does not play any role or plays the role of SP_{app} (the *Car* app in the scenario). In each session, we used different instances of the channels. The complete set of specifications can be found in Appendix B.2.

In Sect. 5.2.2, in relation to the security goal G1$_{MFA}$ (and consequently to G1$_{BA}$), we have described a list of strong and weak assumptions that we have added to the model to constrain the intruder's abilities. Table 7.8 summarizes the security analyses that we have performed to check this

Table 7.8: Analyses performed for G1$_{BA}$ - DigiMat-Lab Scenario.

Analysis	**Strong Asm(s)**	**Weak Asm(s)**	**Atk**
1	all -1	all	Yes
2	all	all -1	No
3	all	all $-m$ $(1 < m \leq 5)$	*

goal.

Regarding the strong assumptions (TA, $CA1_{Var2}$, $CA2_{Var2}$, $CA3_{Var2}$, $CA3_{Var2}$ and AA), we have performed the following analyses:

Analysis 1: We have checked that by removing only one of the six strong assumptions from the model we have a violation of $G1_{BA}$ (i.e. there is an attack). For this analysis, we have thus performed 6 executions of SATMC removing one strong assumption at a time. To provide an example of an attack, Figure 7.13 shows the attack trace deriving from removing $CA1_{Var2}$. In this attack, `i` is able to obtain a *token_SP* (`{idp.citizen.car}_inv(pk(idp))`) and reused it in his smartphone. As a consequence, `i` is authenticated by `car` as the victim (`citizen`). This attack takes advantage of a wrong implementation of the redirection from *Browser* to SP_{app}, in this example `car`. Note that, for the sake of clarity, this figure (and the other figures shown in this section) represents only the significant steps of the attack traces found by the SATMC tool.

Regarding the weak assumptions (BA1, BA2, $UBA1_{Var2}$, $UBA2_{Var2}$, UBA3), we have performed the following analyses that are detailed in Table 7.9:

Table 7.9: Results for $G1_{BA}$ (Analyses 2 and 3).

Removed Weak Asm(s)	**Compromised Factors**				**Atk**
	PIN1	**PIN2**	**cookie**	**CIE**	
BA1	x	✓	✓	x	No
BA2	✓	x	x	x	No
$UBA1_{Var2}$	✓	x	x	x	No
$UBA2_{Var2}$	x	✓	✓	x	No
UBA3	x	x	x	✓	No
$(UBA1_{Var2} \vee BA2) \wedge BA1 \wedge UBA3$	✓	✓	✓	✓	Yes
$(UBA1_{Var2} \vee BA2) \wedge UBA2_{Var2} \wedge UBA3$	✓	✓	✓	✓	Yes

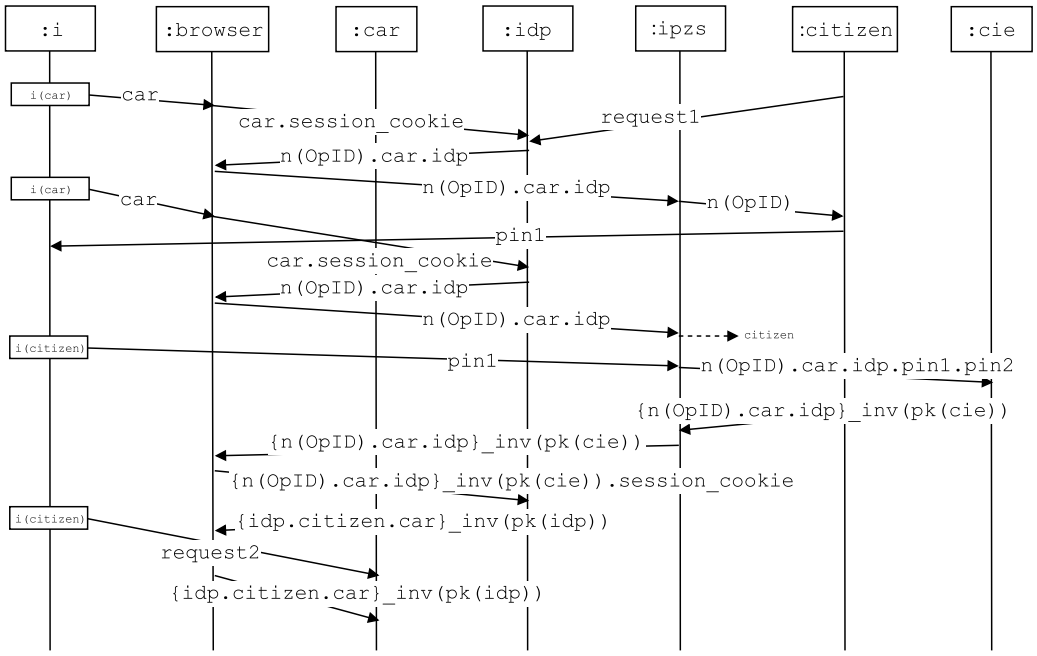


Figure 7.14: Attack trace obtained removing UBA1$_{Var2}$, UBA2$_{Var2}$ and UBA3.

Analysis 2: We have checked that by removing only one of the five weak assumptions from the model, SATMC does not find any attack on the solution (i.e. the intruder is not able to impersonate the user). Indeed, as shown in Table 7.9, by removing only one weak assumption, the intruder obtains only 1 or 2 instance-factors.

Analysis 3: We have checked that by removing specific subsets of weak assumptions it is possible to compromise all the instance-factors, causing a violation of G1$_{BA}$. In Table 7.3, the star (*) denotes that the result can be "yes" or "no" depending on the chosen subset of weak assumptions. The subsets shown in Table 7.4 violate G1$_{BA}$ and result in different attack traces. Figure 7.14 shows the attack trace deriving from removing UBA1$_{Var2}$, UBA2$_{Var2}$ and UBA3 (e.g., a proximity intruder that watches the *PIN1* entered by *Citizen* and then steals the smartphone and the user's *CIE*). In the attack, `i` initiates a session of the protocol on the user smartphone to obtain the

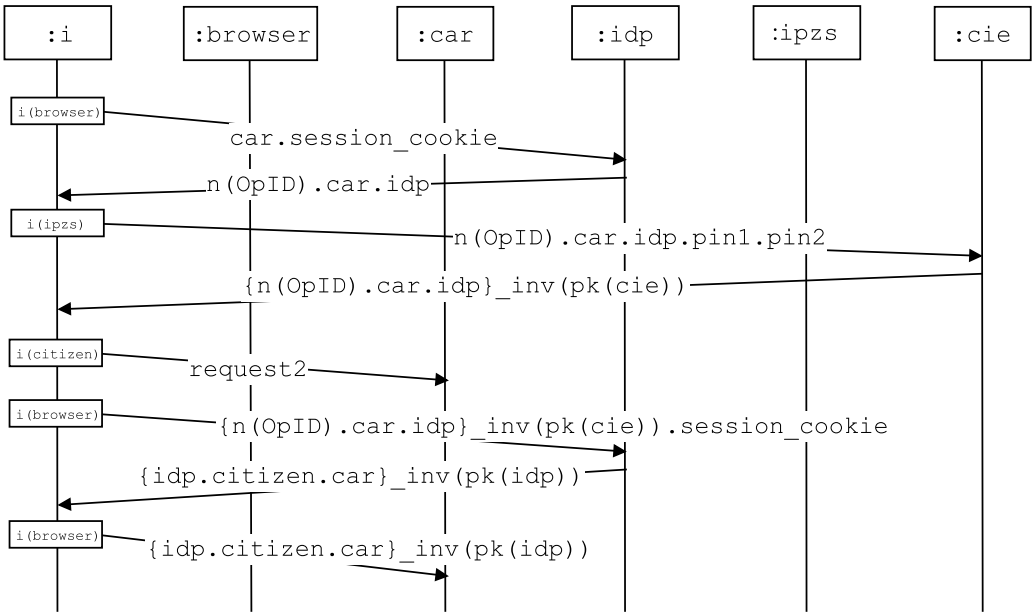


Figure 7.15: Attack trace obtained removing BA1, BA2 and UBA3.

the half PIN code (`pin1`). Then, `i` sends a request (`request2`) to `car` pretending to be `citizen` (indicated as `i(citizen)`). By entering the half PIN code (`pin1`) when requested by `ipzs`, `i` is able to impersonate the `citizen`. Figure 7.15 shows the attack trace deriving from removing BA1, BA2 and UBA3 (e.g., a hacker that discovers *PIN1* typed by *Citizen* using a keylogger, reads *PIN2* and *session_cookie* exploiting a malware installed on the user's smartphone, and asks user to place her *CIE* near the smartphone or steals the *CIE*). In this case, `i` is able to interact with `cie` and `idp`, first obtaining a valid response (`{n(OpID.car.idp)}_inv(pk(cie))`) pretending to be `ipzs` (`i` knows *PIN1* and *PIN2*), and then sending a token request to `idp` pretending to be the browser with user's session (`i` knows `session_cookie`). Finally, `i` uses the obtained *token_SP* (`{idp.citizen.car}_inv(pk(idp))`) to log in as the victim on `car`.

As expected, when checking the solution w.r.t. the security goal G2 — which embodies the OTP properties — under all the (weak and strong)

assumptions, SATMC does not find any attack.

7.3.4 Phase 4: Usability Analysis

The exploitation phase of CIE was designed to augment the usability compared to a classic OTP approach. In particular:

- During the activation phase *IPZS* stores the second part of the user *PIN*. In this way, during the exploitation phase only half *PIN* (four digits) is required.
- The designed solution does not ask *Citizen* to digit the OTP; after the half-*PIN* input, the OTP value is sent to IdP_{IPZS} in a transparent way.
- The designed solution provides a SSO experience: a session cookie is stored in *Browser*. Until the session is valid, *Citizen* has to digit only her half-*PIN* to access *Car* or other federated apps.
- The graphical interface of *IPZS* follows the Italian usability guidelines for PA services (e.g., color choice and font) available in [43].

After the implementation of this solution, we plan to evaluate the usability of the activation and exploitation phases though a pilot. We will ask users to fill two questionnaires dealing with three usability aspects: effectiveness, efficiency and satisfaction. These questionnaires will be based on the ones used for the TreC scenario (see Sect. 7.1.4).

7.4 FIDES - Federation of IdP and Cross Border

FIDES (acronym for Federated IDEntity System) was a two-year activity of the *Privacy, Security and Trust* action line of *EIT Digital* (`www.eitdigital.eu`) with the goal of enabling the secure access of digital and

physical services using the users' digital identity in a federated and cross-border ecosystem.

In the IdM field, a current problem is that *Users* have to manage too many digital identities with different services (social networks, e-commerce or bank), and usually this is complex and can increase security risks (e.g., identity thefts or privacy violations). To avoid these problems, FIDES manages digital identities in a federated way, creating an identity ecosystem that permits the fruition of services in a secure, cross-border (European level) and multi-devices way. The benefits are not only for final *Users*, but also for *SPs*. A challenging task for a *SP* is the integration with *IdPs* that support different IdM protocols (new concepts to learn and implement). In addition, this causes a dramatically increase of development and maintenance costs. With a solution like FIDES, which integrates an identity broker that supports multi-*IdPs*, the management of the digital identities is delegated to the FIDES platform, thus *SPs* have more resources to spend in the creation of innovative services.

Together with the multi-*IdPs* support, a central topic for FIDES is the cross-border electronic identification. As we have described in Sect. 3.2.1, the eIDAS regulation provides an harmonization between the different member states by regulating this aspect. A large-scale of pilots (e.g., STORK 2.0 and FutureID, described in Sect. 8.2) have proven that technical interoperability is possible. In this context, FIDES aim was not to replicate these projects, but design and implement — building on results of these relevant European initiatives — a federated and interoperable IdM platform taking the EU regulations on e-identification into consideration, thus constituting a unified tool to access the EU Digital Single Market.

In addition to the deployment of a multi-*IdPs* and cross-border identity platform, FIDES objectives were: *(i)* the definition of the relevant business model, *(ii)* the delivery of implementation guidelines, and *(iii)* the test of

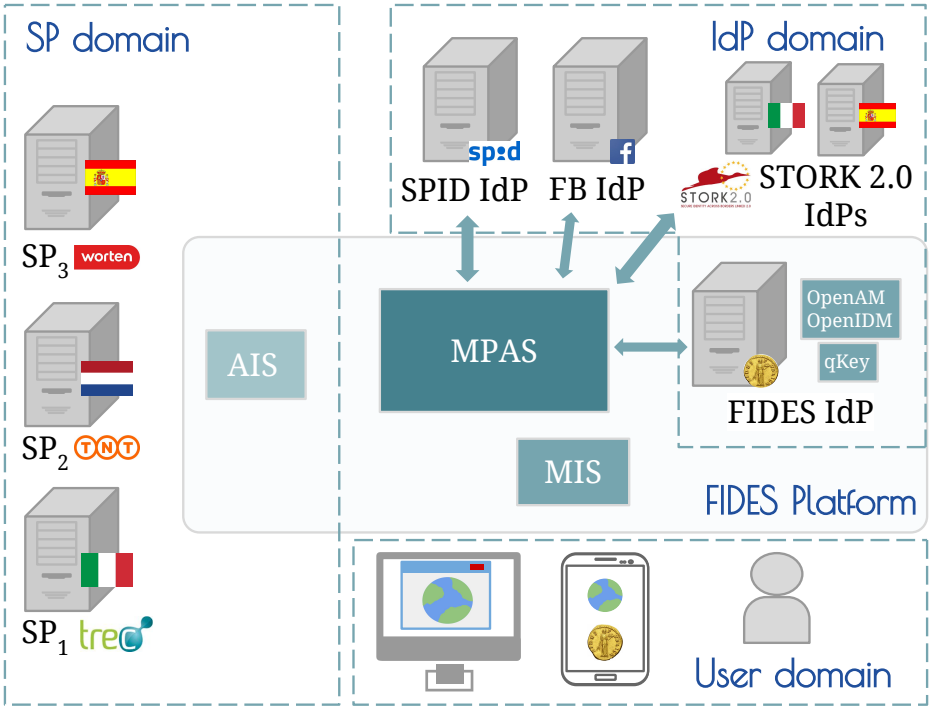


Figure 7.16: FIDES Architecture.

the platform with different pilots. As Security & Trust unit of FBK, we were involved: in the design of the FIDES components for supporting the mobile environment, and in the realization of the Italian pilot relevant to the use of FIDES in mHealth applications.

7.4.1 FIDES Platform

In this section, we illustrate a high-level description of the final architecture of the FIDES platform. Figure 7.16 shows the overall architectural diagram that includes the following main components:

- *STORK 2.0*: technological enabler for cross-border federation (discussed in Sect. 8.2);

- *Fides IDP*: is the FIDES *IdP* and it was build up on ForgeRock Ope-

nAM and OpenIDM technologies[2] and is integrated with qKey, a 2-factor authentication developed by Ubiqu based on software token (described in Sect. 8.3.1).

- *Federated Identity Provider:* FIDES platform supports the integration with the following IdPs: SPID, STORK 2.0 (Italian and Spanish nodes) and Facebook.

- *Service Providers*: applications that want to be part in the FIDES ecosystem. Figure 7.16 shows the three *SPs* involved during the pilot scheme.

- *MPAS (Multi-Provider Authentication Service)*: is an identity broker developed in the context of FIDES that integrates the main outcomes of FutureID components (discussed in Sect. 8.2). Aim of this component is to enhance privacy factors and provide easy integration of new *SPs* and *IdPs* in the FIDES ecosystem. *MIS (Mobile Integration Service)* is an interface of MPAS that support the interaction with mobile applications, and *AIS (Application Integration System)* is a module that *SP* has to integrate to communicate with MPAS.

7.4.2 Phase 1: *AppCtx* Identification

As FIDES is an activity that involves several stockholders, we cannot disclose details about the design and security analysis. However, to have an idea of the effectiveness of our methodology in the FIDES scenario, we describe below the first phase of our methodology.

The FIDES platform can be informally described as a solution that enables *SPs* to integrate into the platform and heritage a list of *IdPs*. In this way, if a *User* has a digital identity with one of the federated *IdPs*,

[2]Details on the ForgeRock open source projects are available at `https://forgerock.org/`.

AppCtx Table		
	Entities	User→**Patient**; SP_{app}→**$TreC_C$**; SP_S→**$TreC_S$** UA→**FIDES IdP_C**; IdP_S→**FIDES IdP_S**; TP_{app}→**qKey app**; TP_S→**qKey server**; IdB→**MPAS**; AP, AS, RS, HWToken→**Null**;
	UA Choice	☐ browser ☑ application
	Data Nature	☐ anonymous ☑ personal ☑ sensitive
	AuthN Aspects	MFA support? ☑ yes ☐ no Session handling? ☑ yes ☐ no
	OTP Choice	☐ TOTP ☑ CR ☐ other

Figure 7.17: FIDES *AppCtx* Table - Italian Pilot.

then she can access a federated *SP* without a further registration. The identification of FIDES *AppCtx* is strictly related to the pilot that involves different SP_S and IdP_S. In FIDES, three pilots were considered:

IT Pilot: the Italian *SP* is *TreC*, a PHR platform of Trentino region (described in Sect. 7.1). Using the FIDES platform *Users* can access their TreC mobile applications logging with the FIDES *IdP* and performing a two-factor authentication with qKey.

NL Pilot: the Dutch *SP* is *TNT innight*, an organization that delivers parcels. Using the FIDES platform a deliveryman can deliver parcels in cars with smart-lockers after the login with the FIDES *IdP* or STORK and using the qKey app to open the car.

ES Pilot: The Spanish *SP* is *Worten*, a company that sales electrical appliance and electronic devices in the retail sector. Using the FIDES platform *Users* can access the online Worten website logging with the Facebook *IdP*. This pilot does not involve mobile applications.

Globally the FIDES platform supports the following authentication aspects: MFA, SSO, Multi-IdPs and Cross-border. In the different pilots, a

subset of these aspects are considered. Figure 7.17 shows the *AppCtx* of the Italian pilot (the only pilot related to IdM for mobile applications).

Chapter 8

Related Work and Initiatives

Those work and solutions that are the most relevant to our scenario have already been discussed in the previous sections. In particular, in Sect. 2.2.4 and Sect. 4.2, we have presented the current IdM protocols (e.g., OAuth for native apps [93]) and solutions (e.g, Facebook [10]) for mobile native applications. In our analysis, we have extracted the key-aspects from these solutions and based our reference model upon them. In addition, a thorough comparison between our reference model and these solutions is discussed in Sect. 5.2.4. Whereas, in this chapter, we will look at our work from a wider perspective by describing different research activities, initiatives and solutions related to IdM. Sect. 8.1 will deal with papers that focus on the security analysis of OAuth 2.0 and OpenID Connect. In Sect. 8.2, we will illustrate some of the current European initiatives on digital identities. Finally, in Sect. 8.3, we will introduce emerging IdM technologies on the market.

8.1 Security Analysis of IdM Protocols

OAuth 2.0 [64] and OpenID Connect [94] have been designed for light-RESTful API services, and are considered the de-facto standards for managing authentication and authorization for web and mobile apps. Much

research has been carried out to discover vulnerabilities in different implementations of these protocols. For instance, Sun et al. [104] analyzed hundreds of OAuth apps focusing on classical web attacks such as Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF). Other studies, such as [108, 103], analyzed the implementations of multi-party web apps via browser-related messages. In the context of mobile apps, a similar work is described in [112], where Yang et al. discovered an incorrect use of OAuth that permits an intruder to login as a victim without the victim's awareness. To evaluate the impact of this attack, they have shown that more than 40% of 600 top-ranked apps were vulnerable.

Although these techniques are useful for the analysis of a specific implementation (as they are able to discover serious security flaws), it is important to perform a comprehensive security analysis of the standard itself. In the context of web apps, Fett et al. [53] performed a formal analysis of the OAuth protocol using an expressive web model (defined in [52]) that describes the interaction between browsers and servers in a real-world set-up. This formal analysis revealed two unknown attacks on OAuth that violate the authorization and authentication properties. A similar analysis is performed for OpenID Connect in [54]. Two other examples of formalizations of OAuth are [33], where the different OAuth flows are modeled in the Applied Pi calculus and verified using ProVerif extended with WebSpi (a library that models web users, apps and intruders), and [97], where OAuth is modeled in Alloy. They found a security vulnerability thanks to which an experienced hacker can retrieve the client credentials by reverse engineering the code when the client credentials were stored in the desktop software. Then, the authors masqueraded a client to harvest sensitive information by using the compromised secret client credentials.

In the formal security analysis described in Sect. 6.2, we use ASLan++ and SATMC. In the past, SATMC has revealed severe security flaws in the

SAML 2.0 protocol [92] and in the variant implemented by Google [30]; by exploiting these flaws a dishonest service provider could impersonate a user at another service provider. Moreover, Yan et al. [111] used ASLan++ and SATMC to analyze four security properties of OAuth: confidentiality, authentication, authorization, and consistency.

The aforementioned formal analyses, however, focus on the web app scenario, whereas in our work we deal with native apps. In [113], Ye et al. used Proverif to analyze the security of a SSO implementation for Android. They applied their approach to the implementation of the Facebook login. Their formal analysis method consists of four stages: *(i)* protocol extraction (analyzing the Facebook SDK and the network traffic); *(ii)* protocol modeling (security properties and attacker models are defined); *(iii)* protocol verification; and *(iv)* vulnerability analysis. The selected tool to perform the formal analysis is Proverif and the protocol is described using the Applied Pi-calculus. As result of this analysis, a vulnerability that exploits super user (SU) permissions is identified. In contrast, our analysis assumes that the user smartphone cannot be rooted. Indeed, if a malicious app is able to obtain a SU permission, then it can set for itself the permission to access all the data stored in the smartphone, compromising all the user data and the tokens of the other apps installed on the rooted smartphone.

8.2 European IdM Initiatives

The theme of digital identity is of critical importance to the European Union (EU), which have promoted many initiatives and guidelines dealing with different dimensions of IdM. We present three major EU initiatives: *STORK 2.0* (Secure idenTity acrOss boRders linKed) [22], *ABC4Trust* (Attribute-based Credentials for Trust) [23] and *Future ID* [56], which deal

with important aspects of a IdM system, such as cross-border, privacy and interoperability.

8.2.1 STORK 2.0

We will start with the analysis of the STORK 2.0 project, which focuses on the cross-border aspect. The STORK platform allows users (physical or legal persons) to use their national electronic ID (eID) to access online services provided by different member states. Thus, the purpose of this project, as well as a central theme in the EU, is the establishment and development of a federated cross-border IdM system.

The STORK platform can be implemented in two different ways. In the first, called "proxy model", the architecture is composed of different instances of the PAN (Personal Area Network) European Proxy Service, called PEPS. In the second, the SP integrates eID tokens using middleware. In this brief overview, we will give more details only on the proxy model, as it better highlights the features and entities involved in the STORK initiative. The model requires a S-PEPS that manages the SP requests, and a C-PEPS that lies in the country which issued the eID that the citizen wants to use. The idea of the authentication process is as follows (see Figure 8.1):

1. a user (Italian citizen) requests the access to a SP in another country (Spain);
2. the SP sends an authentication request to the S-PEPS, declaring the list of attributes required to grant or deny the access to the specific online service;
3. the S-PEPS maps these attributes to the attribute domain of the users country, and sends the modified authentication request to the C-PEPS;

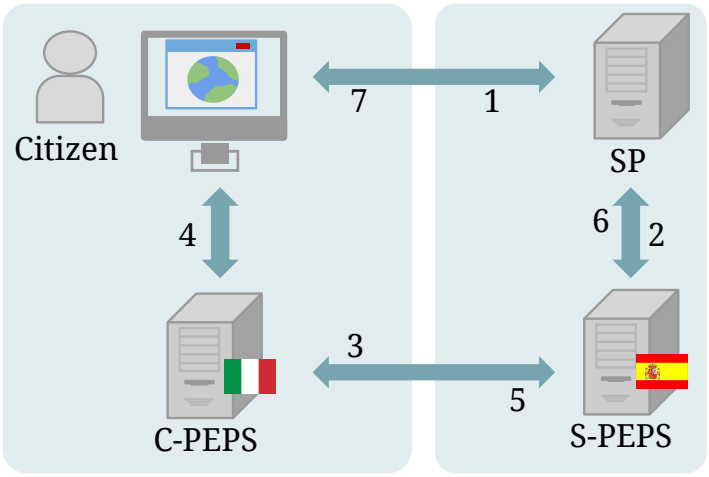


Figure 8.1: PEPS Model (adapted from [80]).

4. the C-PEPS authenticates the user, and
5. sends a SAML assertion back to the S-PEPS;
6. the S-PEPS applies the inverse translation and sends a new assertion with the translated attributes to the SP, which can decide if grant or deny the access to that particular user;
7. if the assertion is valid, the SP returns the protected resource.

8.2.2 ABC4Trust

Another EU initiative is ABC4Trust, which deals with an important aspect of an IdM system: privacy. With the purpose of promoting and simplifying the integration of Privacy-enhanced Attribute-based Credential (Privacy-ABC) techniques, the ABC4Trust project has developed a common architecture, defining the data formats and the interfaces to be used. The Privacy-ABC techniques avoid full identity disclosure: only the information needed by a service provider to grant or deny access are revealed. In addition, the protection of the user privacy is achieved using different fea-

tures, such as the pseudonymity, the use of unlinkability (multi-use), the use and combination of multiple credentials, the minimal and conditional disclosure of attributes.

8.2.3 FutureID

In [99], another EU project, called FutureID is presented. FutureID combines existing eID technologies to provide an eID framework that any service provider can easily integrate into its services. This system provides a security infrastructure to protect the online services, offering a combination of systems with strong authentication and digital signature techniques, and allows us to achieve a user-centric system that manages the identity claims. Concerning the privacy aspect, [99] states that the risk of privacy violations is amplified if the identification process is performed across a complex network of PEPS — Personal Area Network (PAN) European Proxy Service — as is the case of the STORK approach. For this reason, FutureID introduces the *universal authentication service*, which is able to handle different eID cards - and so different protocols - directly.

The illustrated initiatives highlight the European awareness of new systems to control digital identities.

8.3 Emerging IdM Solutions

In this section, we give an overview of emerging IdM technologies. In Sect. 8.3.1, we present the qKey tecnology used in the FIDES EIT Activity to provide a MFA solution for mobile apps. In the MFA context, Sect. 8.3.2 provides an overview of the FIDO ecosystem for MFA solutions. Finally, we give an hint of blockchain-based IdM solutions.

Figure 8.2: qKey Example.

8.3.1 qKey

qKey is a token provider solution that combined with an IdP provides a 2-factor authentication based on a software token. This solution allows a *User* to use her mobile device to provide a MFA without the use of external tokens, such as SIM card or smartcard. This decrease the cost of managing such tokens (e.g., initial physical distribution and revocation in case of updates). The basic idea of the qKey technology is the combination of tree modules: a software that works in a cloud-module (*Hardware Security Module — HSM*); a mobile app (*Authenticate* — available for Android and iOS [1]) that is securely bound in the user device; and a secure bilateral channel between *HSM* and *Authenticate* that allows to send messages in an asynchronous way. The qKey solution is based on a new encryption technology that is certified at the same level of a smartcard (ISO LoA 4). In addition, the design of qKey takes into consideration many security measure to augment the security of the app, such as a daily update of cryptographic key materials.

An example of use of qKey is shown in Figure 8.2, where: *(1) User* clicks on a login button on a website; *(2) User* enters a PIN on the *Authenticate* app installed in her device; *(3)* if the PIN is correct, she is logged in;

In the FIDES EIT Activity (cf. Sect. 7.4), we have applied the qKey technology to provide a MFA solution for the IT pilot (mHealth).

8.3.2 FIDO Alliance

The FIDO (Fast IDentity Online) Alliance [11] has the mission of developing an open ecosystem for standard-based and interoperable authentication solution. It provides two specifications on two different authentication experiences: passwordless experience (*Universal Authentication Framework, UAF*) and second factor experience (*Universal Second Factor, U2F*). Both specifications are based on standard public key cryptography and require a "registration" operation (performed each time *User* wants to login to a *SP* she has never accessed before). The entities involved are: a *User* who wants to access a service (*SP*); FIDO Client ($FIDO_C$) and FIDO Server ($FIDO_S$) that manage the *SP* registration and *User* authentication.

Universal Authentication Framework (UAF)

In UAF protocol, $FIDO_C$ is usually played by an application that stores multiple private keys (one for each *SP*) and protects them with a local authentication methods (e.g., PIN or fingerprint). UAF protocol supports two operations:

Registration: *User* is asked to provide the local authentication method to access $FIDO_C$ and confirm her registration; a key pair is then generated by $FIDO_C$ (the public key is communicated to $FIDO_S$ and the private key is stored locally).

Authentication: *User* accesses *SP* by only providing her local authentication method.

Key points: one local authentication method for all *SP* and multiple (independent) keys for different *SPs*.

Universal Second Factor (U2F)

U2F also involves the usage of an hardware device, called *FIDO U2F* (e.g., Yubico key), for providing an extra identity proof to basic authentication. U2F protocols support two operations: registration and authentication. Both operations consist of the following phases:

Setup: $FIDO_C$ contacts *SP* and obtains a challenge. Using the challenge (and possibly other data obtained from *SP* and/or prepared by $FIDO_C$ itself), $FIDO_C$ prepares a request message for the *FIDO U2F*.

Processing: $FIDO_C$ sends the request message to *FIDO U2F*, and *FIDO U2F* performs some cryptographic operations on the message, creating a response message. This response message is sent to $FIDO_C$.

Verification: $FIDO_C$ transmits the *FIDO U2F*'s response message, along with other data necessary for *SP* to verify the response, to *SP*. *SP* then processes the response and verifies its correctness. A correct registration response will cause *SP* to register a new public key for *User*, while correct authentication response will cause *SP* to accept that $FIDO_C$ is in possession of the corresponding private key.

Key Points: integrates existing authentication mechanisms, *U2F Token* generates and stores keys for different *SPs*, possibility to communicate with different channels (e.g., USB, NFC and Bluetooth).

Applications

The FIDO Alliance lists many *FIDO U2F* from different vendors such as: Bluink (Bluink Key), Century Longmai Technology (mFIDO U2 token), Hypersecu Information Systems (HyperFIDO Mini), Vasco (DIGIPASS SecureClick) and Yubico (YubiKey 4, 4C, NEO, 4 Nano, FIDO U2F Security Key).

About these solutions, *YubiKey NEO* [114] is one of the most attractive for mobile. It is a token device that supports OTPs and, by integrating a NFC (Near Field Communication) technology, it can be used to provide a second-factor also in the mobile context. Compared to this product, our solution provides a multi-factor authentication solution for native mobile apps without requiring an additional device.

8.3.3 Blockchain for IdM

The success of the Bitcoin [2] system based on distributed ledger technology (DLT) — sometimes referred to as blockchain technology — has raised interest in studying fully decentralized IdM solutions. An example is the definition of DKMS (Decentralized Key Management System) [5]. DKMS is a new approach based on DLTs that manages cryptographic keys without depending on centralized authorities. In DKMS — in contrast to conventional PKI (public key infrastructure) architecture — the initial trust is a distributed ledger that supports a new form of identifier, called DID (Decentralized IDentifier), used to verify digital identities that are *self-sovereign* (i.e. fully controlled by the user without IdPs or certificate authority). A DID is generated cryptographically and points to a DDO (DID descriptor object) that contains the associated public verification key(s) and a pointer to a distributed ledger supporting peer-to-peer interactions with the identity owner. Trust can be established in two ways: challenge-response for a real-time verification of the public key; or exchange of identity attributes signed by a trusted peer who can be verified against the distributed ledger. Some of the benefits of using DKMS are: *(i)* no single point of failure as there is no centralized authority; *(ii)* interoperability as any apps can perform a key exchange without depending on proprietary software or services; and *(iii)* the key recovery process built directly into the infrastructure.

A wide range of open initiatives and organization are focusing on distributed IdM. An example is Sovrin [19], a technology based on DKMS that allows for the creation of DID stored on a public permissioned ledger run by a group of trusted organizations spread across the globe.

Chapter 9

Conclusions and Future Work

Although many standards and initiatives have been developed to manage digital identity processes, there remains much work that still needs to be addressed, especially for securing mobile solutions. As described in [37], it is a good practice to incorporate privacy and security in the solution not as an afterthought, but rather, by design. Even though security-by-design and privacy-by-design are well-accepted concepts, their application is not easy and depends on many aspects. In this regard, we have defined a specific design methodology that can be used by IdM designers for the design and security assessment of authentication and authorization solutions for mobile apps.

A central aspect of our methodology is the definition of a reference model (*mID(OTP)*) that resulted from a a rational reconstruction of Facebook mobile IdM solution and an analysis of the OAuth specification for native applications. We called it *mID(OTP)* to highlight the dual goal that our model pursues: *(i)* "*mID*" represents the management of identities for native mobile apps, providing a Single Sign-On (SSO) experience, while *(ii)* "(*OTP*)" represents the optional establishment of a Multi Factor Authentication (MFA) which is parametric on the OTP generation approach. In our analysis, we have detailed the OTP generation box for TOTP and

challenge-response approaches. By following our methodology, an IdM designer is guided in the customization of *mID(OTP)* and in the security and usability analysis of the resulting customized IdM solution. In particular, our methodology consists of four phases: *(i)* an identification of the application context by specifying the entities that are involved in the scenario and the authentication and authorization aspects (e.g., MFA or SSO experience support) to take into consideration; *(ii)* the customization of *mID(OTP)*, as a result of which an IdM designer will obtain a protocol flow description, plus a list of security assumptions and goals instantiated for his solution; *(iii)* a security analysis and *(iv)* a usability analysis (e.g., by using ad-hoc questionnaires). Regarding the security analysis (Phase 3), our methodology supports analyses ranging from semi-formal to formal. For the former, our methodology provides an informal description of the adversarial model that can be customized by the IdM designer; the adversarial model, together with the output of Phase 2 (MSC, assumptions and goals), are the starting point to argue whether the protocol satisfies the specified properties. For the latter, an IdM designer is required to specify the protocol flow and the security properties using one of the available formal specification languages for the description of cryptographic and browser-based protocol, and verify the security property violations using an automated tool for protocol analysis. For this analysis, our methodology provides a mapping from the informal protocol description (output of Phase 2) to a formal specification using ASLan++ [32], a formal specification language developed in the context of the AVANTSSAR EU-founded project [26]. In general, provided that the IdM designer performs the mapping (gray box of Phase 3 of Figure 4.2) between the informal specification (output of Phase 2) to the selected formal language, our methodology is parametric on the formal specification language. In our methodology, we require the use of a formal analysis when the solution deals with sensitive

data or the overall risk is high. Finally, for the last phase (usability analysis) our methodology provides a list of usable aspects to consider during the design and a questionnaire template for evaluating user satisfaction.

Summarizing, we have defined a methodology for the design and security assessment of mobile IdM solutions, and made the following contributions:

- *for the design:* the flow, security assumptions and goals of a reference model (called *mID(OTP)*) that provides a SSO experience and a MFA support for native apps using OTPs;
- *for the security assessment:* an adversarial model for the semi-formal approach and a formal mapping (from an informal description to ASLan++) of *mID(OTP)* flow, assumptions and goals (also for a MFA solution) for the formal approach;
- *for the usability analysis:* a set of usability considerations and a questionnaire template for evaluating user satisfaction.

To validate our methodology and the corresponding reference model, we have applied it to four different real-world scenarios that represent different functional and usability requirements; for two of them, we have performed a formal security analysis detailing the OTP approach.

9.1 Future Work

The reference model we have presented, as well as the formal specification and analysis that we have given, can be generalized quite straightforwardly to other use-cases, which we are currently doing. In addition, we plan to extend our methodology on different ways.

A first important improvement, is to provide — together with the custom design (output of Phase 2) — the corresponding implementation code. In this way, an IdM designer could perform two different security analyses:

on the design protocol (using one of the approaches described in Chapter 6), and on the implementation (e.g., using penetration testing tools). In this way, we will highlight the link between design and implementation mechanisms described in Sect. 4.1.2, and we will extend the security analysis by considering implementation aspects that in this work are assumed as well-implemented.

A second improvement of our methodology will be to extend the MFA support by analyzing and modeling other authentication factors, such as other OTP approaches or biometric traits (e.g., the use of fingerprint instead of the PIN) that are usually consider as more usable for the final user.

Finally, we are planning to automate same phases of our methodology, e.g., creating an eclipse plugin that performs the formal MFA analysis (launching SATMC with different assumptions and goals) given in input an informal description of the attacker types (e.g., proximity attacker or hacker).

Bibliography

[1] Authenticate app of Ubiqu. https://play.google.com/store/apps/details?id=com.ubiqu.eid.

[2] Bitcoin. https://bitcoin.org/.

[3] City Service Development Kit. http://www.citysdk.eu.

[4] Cloud Foundry. https://www.cloudfoundry.org/.

[5] Decentralized Key Management System. https://github.com/WebOfTrustInfo/rebooting-the-web-of-trust-spring2017/blob/master/topics-and-advance-readings/dkms-decentralized-key-mgmt-system.md.

[6] Design definition. https://www.macmillandictionary.com/dictionary/british/design_1.

[7] Design thinking. http://whatis.techtarget.com/definition/design-thinking.

[8] Digital Identity Guidelines - Authentication and Lifecycle Management. https://pages.nist.gov/800-63-3/sp800-63b.html.

[9] DPCM of 24 October 2014, Sistema Pubblico per la gestione dell'Identità Digitale (SPID). http://www.agid.gov.it/agenda-digitale/infrastrutture-architetture/spid.

[10] Facebook. Getting started with the Facebook SDK for Android, January 2015. https://developers.facebook.com/docs/android/getting-started/facebook-sdk-for-android/.

[11] FIDO. https://fidoalliance.org/about/what-is-fido/.

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

[13] Getting Started with OAuth 2.0. http://it-ebooks.info/read/664/.

[14] macOS High Sierra Bug Lets Anyone Gain Root Access Without a Password. https://thehackernews.com/2017/11/mac-os-password-hack.html.

[15] Oracle's Smart City Platform. http://www.oracle.com/us/industries/public-sector/national-local-government/city-platform/index.html.

[16] Profiles for the OASIS. Security Assertion Markup language (SAML) V2.0. http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf.

[17] Scoppia il caso dei dati fiscali altrui liberamente consultabili. Ma Sogei recidiva. http://www.dday.it/redazione/24301/scoppia-il-caso-dei-dati-fiscali-altrui-liberamente-consultabili-ma-sogei-e-recidiva.

[18] Smartphone Secure Development Guidelines for App Developers. https://www.enisa.europa.eu/publications/smartphone-secure-development-guidelines.

[19] Sovrin identity for all. https://sovrin.org/.

[20] OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data, 1980.

[21] The OECD Privacy Framework. http://www.oecd.org/sti/ieconomy/oecd_privacy_framework.pdf, 2013.

[22] STORK 2.0. Secure Identity Across Borders Linked 2.0. https://www.eid-stork2.eu/.

[23] ABC4Trust. Attribute-based Credentials for Trust. https://abc4trust.eu/.

[24] AgiD. Avviso nr 4. Livelli di servizio minimo per funzionalit omogenee. http://www.agid.gov.it/sites/default/files/documentazione/spid-avviso-n4-livelli-servizio-minimo-funz-omogenee.pdf, 2016.

[25] Android. Handling Android App Links. https://developer.android.com/training/app-links/index.html, 2017.

[26] A. Armando, W. Arsac, T. Avanesov, M. Barletta, A. Calvi, A. Cappai, R. Carbone, Y. Chevalier, L. Compagna, J. Cuéllar, G. Erzse, S. Frau, M. Minea, S. Mödersheim, D. V. Oheimb, G. Pellegrino, S. E. Ponta, M. Rocchetto, M. Rusinowitch, M. T. Dashti, M.Turuani, and L. Viganò. The AVANTSSAR platform for the automated validation of trust and security of service-oriented architectures. In *Proceedings of the 18th international Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'12)*, pages 267–282, 2012.

[27] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, P. C. Heám, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. v. Oheimb, M. Rusinowitch, J. Santiago,

and M. Turuani. The AVISPA tool for the automated validation of internet security protocols and applications. In *Proceedings of 17th international conference on Computer Aided Verification (CAV'05)*, 2005.

[28] A. Armando, R. Carbone, and L. Compagna. LTL model checking for security protocols. *Journal of Applied Non-Classical Logics*, 19(4):403–429, 2009.

[29] A. Armando, R. Carbone, and L. Compagna. SATMC: a SAT-based Model Checker for Security-critical Systems. In *Proceedings of the 20th international Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'14)*, pages 31–45, 2014.

[30] A. Armando, R. Carbone, L. Compagna, J. Cuéllar, and L. Tobarra. Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for google apps. In *FMSE '08:Proceedings of the 6th ACM workshop on Formal methods in security engineering, ACM, New York, NY, USA*, pages 1–10, 2008.

[31] A. Armando, R. Carbone, and L. Zanetti. Formal Modeling and Automatic Security Analysis of Two-Factor and Two-Channel Authentication Protocols. In *Proceedings of the 7th NSS*, pages 728–734, 2013.

[32] AVANTSSAR Project. Deliverable D2.3 (update) ASLan++ specification and tutorial. `http://www.avantssar.eu/pdf/deliverables/avantssar-d2-3_update.pdf`, 2008.

[33] C. Bansal, K. Bhargavan, and S. Maffeis. Discovering Concrete Attacks on Website Authorization by Formal Analysis. In *25th IEEE Computer Security Foundations Symposium (CSF'12)*, pages 247–262, 2012.

[34] B. Blanchet, B. Smyth, and V. Cheval. ProVerif 1.88: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial. http://www.bensmyth.com/files/ProVerif-manual-version-1.88.pdf, 2013.

[35] G. W. Van Blarkom, J. J. Borking, and J.G.E. Olk. Handbook of Privacy and Privacy-Enhancing Technologies (The Case of Intelligent Software Agents). https://www.andrewpatrick.ca/pisa/handbook/handbook.html, 2003.

[36] CAD. Codice dell'Amministrazione Digitale - D.Lgs.n. 82/2005. http://www.altalex.com/documents/codici-altalex/2014/06/20/codice-dell-amministrazione-digitale, 2014.

[37] A. Cavoukian and M. Chanliau. Privacy and Security by Design: A Convergence of Paradigms, 2013.

[38] D. Chell, T. Erasmus, S. Colley, and O. Whitehouse. *The Mobile Application Hacker's Handbook*. Wiley Formats, 2015.

[39] E. Chen, Y. Pei, S. Chen, Y. Tian, R. Kotcher, and P. Tague. OAuth Demystified for Mobile Application Developers. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2014.

[40] E. Chin, A. P. Felt, K. Greenwood, and D. Wagner. Analyzing inter-application communication in android. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, pages 239–252, 2011.

[41] Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems: Concepts and Design (4th Edition) (International Computer Science)*.

Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[42] G. Danezis, J. Domingo-Ferrer, M. Hansen, J.H. Hoepman, D. Le Métayer, R. Tirtea, and S. Schiffner. ENISA report: Privacy and Data Protection by Design - from policy to engineering, 2014.

[43] Design Italia. Designer dalla parte dei cittadini. https://designers.italia.it/, 2017.

[44] D. Dolev and A. Yao. On the Security of Public-Key Protocols. In *IEEE Transactions on Information Theory.*, page 2(29), 1983.

[45] EC. A Digital Single Market Strategy for Europe. http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A52015DC0192, 2015.

[46] ECB – European Central Bank. Final guidelines on the security of internet payments. https://www.eba.europa.eu/documents/10180/934179/EBA-GL-2014-12+%28Guidelines+on+the+security+of+internet+payments%29.pdf/f27bf266-580a-4ad0-aaec-59ce52286af0, 2014.

[47] EU Commission. Common Terminological Framework for Interoperable Electronic Identity Management Consultation paper. http://ec.europa.eu/information_society/activities/ict_psp/documents/eid_terminology_paper.pdf, 2005.

[48] European Data Protection. Directive 95/46 EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and the free movement of such data. http://eur-lex.europa.eu/legal-

content/EN/TXT/?uri=CELEX:31995L0046. The "General Data Protection Regulation" (Regulation EU 2016/679, http://www.eugdpr.org), adopted in April 2016, will supersede the Data Protection Directive and will be enforceable starting on 25 May 2018.

[49] European Parliament. Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures. http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:31999L0093&from=EN.

[50] European Parliament. Regulation 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC. http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32014R0910&from=EN.

[51] Facebook. Manually build a login flow. https://developers.facebook.com/docs/facebook-login/manually-build-a-login-flow/, 2016.

[52] D. Fett, R. Küsters, and G. Schmitz. An Expressive Model for the Web Infrastructure: Definition and Application to the BrowserID SSO System. In *Proceedings of the 35th IEEE Symposium on Security and Privacy (S&P)*, pages 673–688. IEEE Computer Society, 2014.

[53] D. Fett, R. Küsters, and G. Schmitz. A Comprehensive Formal Security Analysis of OAuth 2.0. In *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1204–1215. ACM, 2016.

[54] D. Fett, R. Küsters, and G. Schmitz. The Web SSO Standard OpenID Connect: In-Depth Formal Security Analysis and Security Guide-

lines. In *Proceedings of the 30th Computer Security Foundations Symposium (CSF)*. IEEE Computer Society, 2017.

[55] G. Fioroni, M. Pistore, S. Ranise, G. Sciascia, M. Trainotti, F. Amigoni, L. Caporusso, F. Gleria, and A. Maffeis. Smart Government - Toward an Innovative Concept of a "One-Stop Shop" for Interactive Online Services. In *IEEE Smart Cities Inaugural Workshop*, 2014.

[56] FutureID. Shaping the Future of Electronic Identity. `http://www.futureid.eu/`.

[57] Garante Privacy. Personal Data Protection Code. Legislative Decree no. 196 of 30 June 2003. `http://194.242.234.211/documents/10160/2012405/Personal+Data+Protection+Code+-+Legislat.+Decree+no.196+of+30+June+2003.pdf`, 2003.

[58] Gemalto. Poor Internal Security Practices Take a Tool. `http://breachlevelindex.com/assets/Breach-Level-Index-Report-H1-2017-Gemalto.pdf`, 2017.

[59] Gigya. The Landscape of Customer Identity: Facebook Slides Again. `https://www.gigya.com/blog/the-landscape-of-customer-identity-facebook-slides-again/`.

[60] Google. Google Authenticator. `https://support.google.com/accounts/answer/1066447?hl=en`.

[61] P. Guarda. Data Protection, Information Privacy, and Security Measures: An Essay on the European and the Italian Legal Framework. `https://ssrn.com/abstract=1517449`, 2008.

[62] P. Guarda and N. Zannone. Towards the development of privacy-aware systems. *Information and Software Technology*, 51(2):337 – 350, 2009.

[63] IETF. TOTP: Time-Based One-Time Password Algorithm. `https://tools.ietf.org/html/rfc6238`, 2011.

[64] IETF. The OAuth 2.0 Authorization Framework. `http://tools.ietf.org/html/rfc6749`, 2012.

[65] IETF. The OAuth 2.0 Authorization Framework: Bearer Token Usage. `https://tools.ietf.org/html/rfc6750`, 2012.

[66] IETF. OAuth 2.0 Threat Model and Security Considerations. `https://tools.ietf.org/html/rfc6819`, 2013.

[67] IETF. JSON Web Token (JWT). `https://tools.ietf.org/html/rfc7519`, 2015.

[68] IETF. OAuth 2.0 Token Introspection. `https://tools.ietf.org/html/rfc7662`, 2015.

[69] IETF. Proof Key for Code Exchange by OAuth Public Clients. `https://tools.ietf.org/html/rfc7636`, 2015.

[70] GSMA Intelligence. Definitive data and analysis for the mobile industry. `https://gsmaintelligence.com/`.

[71] International Organization for Standardization. ISO/IEC 24760-1:2011, A framework for identity management, Part 1: Terminology and concepts. `https://www.iso.org/obp/ui/#iso:std:iso-iec:24760:-1:ed-1:v1:en`.

[72] International Organization for Standardization. ISO/IEC 29115:2013, Information technology – Security techniques

– Entity authentication assurance framework. https://www.iso.org/obp/ui/#iso:std:iso-iec:29115:ed-1:v1:en.

[73] International Organization for Standardization. ISO/IEC 29128:2011, Information technology – Security techniques – Verification of cryptographic protocols. https://www.iso.org/standard/45151.html.

[74] International Organization for Standardization. ISO/IEC 9241-210:2010, Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems. https://www.iso.org/obp/ui/#iso:std:iso:9241:-210:ed-1:v1:en.

[75] iOS. Configuring Your Xcode Project for Distribution. https://developer.apple.com/library/content/documentation/IDEs/Conceptual/AppDistributionGuide/ConfiguringYourApp/ConfiguringYourApp.html, 2017.

[76] iOS. Inter-App Communication. https://developer.apple.com/library/content/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/Inter-AppCommunication/Inter-AppCommunication.html#//apple_ref/doc/uid/TP40007072-CH6-SW2, 2017.

[77] iOS. Universal Links for Developers. https://developer.apple.com/ios/universal-links/, 2017.

[78] T. Khorana. Fake Facebook Phishing Attack. https://sites.google.com/site/mobilesecuritylabware/9-mobile-phishing/post-lab-activities/lab-1-fake-facebook-phishing-attack, n.d.

[79] L. Lamport. Password Authentication with Insecure Communication Communications. *Commun. ACM*, 24(11):770–772, 1981.

[80] H. Leitold and B. Zwattendorfer. Stork: Architecture, implementation and pilots. In *ISSE 2010 Securing Electronic Business Processes*, pages 131–142. 2011.

[81] J.R. Lewis. IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use. *International Journal of Human-Computer Interaction*, pages 57–78, 1995.

[82] G. Lowe. A Hierarchy of Authentication Specifications. In *Proceedings of the 10th IEEE Workshop on Computer Security Foundations*, 1997.

[83] P. Madsen. Mobile OS Developments & Native Application Authentication. `https://www.pingidentity.com/en/blog/2015/06/19/mobile_os_developments_native_application_authentication.html`, 2015.

[84] P. Madsen. NAPPS has left the building (but is still on the front lawn). `https://www.pingidentity.com/en/blog/2015/07/22/napps_has_left_the_building_but_is_still_on_the_front_lawn.html`, 2015.

[85] Ministero dell'Interno. Carta di Identità Elettronica. `http://www.cartaidentita.interno.gov.it/`, 2017.

[86] Ministero dell'Interno and IPZS. Carta di Identità Elettronica CIE 3.0 - Specifiche Chip. `http://www.cartaidentita.interno.gov.it/wp-content/uploads/2016/07/cie_3.0_-_specifiche_chip.pdf`, 2017.

[87] S. Mödersheim and L. Viganò. Secure Pseudonymous Channels. In *Computer Security – ESORICS 2009: 14th European Symposium on Research in Computer Security.*, pages 337–354, 2009.

[88] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.

[89] Network Working Group. Internet Security Glossary, Version 2. `https://tools.ietf.org/html/rfc4949`.

[90] NIST. Special Publication 800-63-3: Digital Identity Guidelines. `http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-3.pdf`.

[91] NIST. Special Publication 800-63b: Digital Identity Guidelines (Authentication and Lifecycle Management). `http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf`.

[92] OASIS. SAML V2.0 technical overview. `http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-cd-02.pdf`, 2005.

[93] OAuth Working Group. OAuth 2.0 for Native Apps. `https://tools.ietf.org/html/draft-ietf-oauth-native-apps-09`, 2016.

[94] OIDF. OpenID Connect Core 1.0. `http://openid.net/specs/openid-connect-core-1_0.html`, 2014.

[95] OIDF. OpenID Connect Native Application Token Agent Core 1.0. `https://openid.bitbucket.io/draft-native-application-agent-core-01.html`, 2014.

[96] D. Otway and O. Rees. Efficient and timely mutual authentication. *SIGOPS Oper. Syst. Rev.*, 21(1):8–10, 1987.

[97] S. Pai, Y. Sharma, S. Kumar, R. M. Pai, and S. Singh. Formal Verification of OAuth 2.0 Using Alloy Framework. pages 655–659, 2011.

[98] G. Pascuzzi. In *Il diritto nell'era digitale*, page 47, 2016.

[99] H. Ronagel, J. Camenisch, L. Fritsch, T. Gross, D. Houdeau, D. Hhnlein, A. Lehmann, and J. Shamah. FutureID - Shaping the Future of Electronic Identity. In *Proceedings of Annual Privacy Forum 2012, LNCS (Springer, 2012).*

[100] M. Shehab and F. Mohsen. Towards Enhancing the Security of OAuth Implementations in Smart Phones. In *IEEE International Conference on Mobile Services (MS)*, pages 39–46, 2014.

[101] T. Shields. Why Security Depends on Usability and How to Achieve Both. `https://www.darkreading.com/risk/why-security-depends-on-usability----and-how-to-achieve-both/a/d-id/1330485`, 2017.

[102] W. Stallings and L. Brown. *Computer Security: Principles and Practice.* Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2014.

[103] A. Sudhodanan, A. Armando, R. Carbone, and L. Compagna. Attack Patterns for Black-Box Security Testing of Multi-Party Web Applications. In *23nd Annual Network and Distributed System Security Symposium (NDSS)*, 2016.

[104] S. Sun and K. Beznosov. The Devil is in the (Implementation) Details: An Empirical Analysis of OAuth SSO Systems. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS'12)*, 2012.

[105] Surfnet. OpenID Connect for SURFconext. Assessment of the OpenID Connect protocol for Federations of Higher Education and Research. https://blog.surfnet.nl/wp-content/uploads/2013/04/SURFnet-OpenID-Connect-1.1-.pdf, 2012.

[106] T. Tullis and B. Albert. *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics*. Morgan Kaufmann Publishers Inc., 2013.

[107] M. Vanhoef and F. Piessens. Key Reinstallation Attacks: Forcing Nonce Reuse in wpa2. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1313–1328, 2017.

[108] R. Wang, S. Chen, and X. Wang. Signing Me onto Your Accounts through Facebook and Google: A Traffic-Guided Security Study of Commercially Deployed Single-Sign-On Web Services. In *IEEE Symposium on Security and Privacy (SP)*, pages 365–379, 2012.

[109] S. Warren and L. Brandeis. The Right to Privacy. In *The Harvard Law Review Association*, pages 193–220, 1890.

[110] We Are Social and Hootsuite. Digital in 2017 Global overview. https://wearesocial.com/special-reports/digital-in-2017-global-overview.

[111] H. Yan, H. Fang, C. Kuka, and H. Zhu. Verification for oauth using aslan++. In *Proceedings of 16th HASE*, pages 76–84, 2015.

[112] R. Yang, W. C. Lau, and T. Liu. Signing into one billion mobile app accounts effortlessly with oauth2.0. 2016.

[113] Q. Ye, G. Bai, K. Wang, and J. S. Dong. Formal analysis of a single sign-on protocol implementation for android. In *Proceedings of the 20th ICECCS*, pages 90–99, 2015.

[114] Yubico. YubiKey NEO. `https://www.yubico.com/products/yubikey-hardware/yubikey-neo`.

Appendix A

Assumptions and G1$_{BA}$

We report here two claims that informally prove that all the assumptions of Sect. 5.2.2 allow to prevent a violation of the security goal G1$_{BA}$.

Claim 1. *If at least one of the following properties is not satisfied then also G1$_{BA}$ does not hold:*

(P1) *Secrecy of user's IdP credentials (among User, UA and IdP$_S$).*

(P2) *Secrecy of token_IdP or session_cookie (among UA and IdP$_S$).*

(P3) *Secrecy of token_SP (among SP$_{app}$, UA and IdP$_S$).*

Where the secrecy of a message M among a set of entities is guaranteed if, and only if, only these entities can know M.

Proof. If P1 does not hold then an attacker can trivially use the victim's IdP credentials to authenticate in SP_{app} as the victim, thus violating G1$_{BA}$. If P2 does not hold then an attacker can use the user session token/cookie to authenticate in SP_{app} as the victim. Finally, if P3 does not hold then G1$_{BA}$ is violated. Indeed, as described in [39], if a malicious agent (using an app) has read a victim's *token_SP* from the victim's smartphone, then he can use this token (linked with the victim's identity) on his smartphone to authenticate at SP_{app} as the victim. □

Claim 2. *BA1, BA2, TA, $CA1_{Var1}$, $CA2_{Var1}$, $CA3_{Var1}$, $UBA1_{Var1}$, $UBA2_{Var1}$, AA of Sect. 5.2.2 allow to prevent a violation of $G1_{BA}$, similarly for Var2 assumptions.*

Proof. If BA1 does not hold, it means that a malicious app can read the values stored by *UA* and SP_{app} (e.g., *token_IdP*, *session_cookie* and *token_SP*), thus we have a violation of P2 and P3. If BA2 does not hold, an attacker can read the input entered by *User* (e.g., user credentials or PIN) thus violating P1.

If TA does not hold, then *IdTP* could be malicious and SP_{app} may obtain a fake identity assertion and login the wrong *User* (violation of $G1_{BA}$).

We require $CA1_{Var1}$ and $CA2_{Var1}$, otherwise *token_SP* could be sent to an untrusted entity (violation of P3). If $CA3_{Var1}$ does not hold, then *token_IdP* could be sent to an untrusted site (violation of P2). If $UBA1_{Var1}$ is not valid, a *User* could enter an authentication factor in a malicious app. For example, during the activation phase, a *User* could enter her IdP's credential in an untrusted entity (violation of P1). While, if $UBA2_{Var1}$ is not valid, this means that a person different from *User* could communicate with that particular installation of *IDOTP*. This is possible only if the user's smartphone is stolen. Having the smartphone, an attacker can extract the stored token values violating P2 and P3. Finally, we need AA, to prevent that a malicious app knows the user's credentials. With the credentials the attacker can perform an activation phase and obtain the corresponding tokens or cookies (violation of P1 and P2). The proof is similar for the security assumptions of *mID(OTP) Var2*. □

Appendix B

ASLan++ Files and SATMC Tool

To perform our security test we have used SATMC model checker tool (Version 3.5.7) launched within Eclipse using the STIATE plugin (Version 1.0.0.1). Links to download the SATMC tool and the STIATE plugin, together with the ASLan++ specification and the original attack trace charts are available on the website `https://sites.google.com/view/idotp-formal-specification`. In the following, we report only the ASLan++ specification for TreC and CIE scenarios.

B.1 TreC ASLan++

We report here the complete ASLan++ specification for the TreC scenario.

```
1  specification otppat
2  channel_model ACM
3
4  entity Environment{
5
6  types
7  key_hash < text;
8  seed < text;
9  time < text;
10 pin < symmetric_key;
11 token < text;
```

```
12
13 symbols
14 trec, optpat, adc, patient: agent;
15 keyhash, keyhash_i: key_hash;
16 nonpublic token_idp: token;
17 nonpublic seedotp: seed;
18 ctime: time;
19 metadata, metadata_i: text;
20 nonpublic pinUser: pin;
21 request1, request2:text;
22
23 ch_t2o, ch_o2t, ch_o2a, ch_a2o, ch_p2o, ch_o2p, ch_p2t, ch_t2p: channel;
24 ch_i2o, ch_o2i, ch_i2t, ch_t2i, ch_t2o1, ch_o2t1, ch_o2a1, ch_a2o1: channel;
25 ch_p2o1, ch_o2p1, ch_p2t1, ch_t2p1: channel;
26
27 nonpublic trustedSPs(agent): agent*key_hash set;
28 nonpublic usersDB(agent): agent*seed*text set;
29 nonpublic pinIDOTP(agent): pin set;
30 nonpublic metadataDB(agent): key_hash*text set;
31
32 infoMobile: agent*key_hash set;
33 userconsent: fact;
34 noninvertible otp_generation(seed,time): text;
35
36 %%%%%%% SESSION %%%%%%%
37 entity Session(TreC, OTPPAT, ADC, Patient: agent, Ch_T2O, Ch_O2T, Ch_O2A,
38 Ch_A2O, Ch_P2O, Ch_O2P, Ch_P2T, Ch_T2P: channel, Seed: seed, Token_IDP: token,
39  CTime: time, PIN: pin, Request: text){
40
41 %%%% Patient %%%%
42 entity User(Actor, OTPPAT, TreC: agent, Ch_P2O, Ch_O2P, Ch_P2T, Ch_T2P:
43 channel, PIN: pin, Request: text){
44
45 symbols
46 PINRequest: text;
47
48 body{%of user
49 Actor -Ch_P2T-> TreC: SP_authn_U_on_Request:(Request); %STEP A1
```

```
50          select{on(OTPPAT -Ch_O2P-> Actor: ?PINRequest):{ %STEP A6.a
51          Actor -Ch_P2O-> OTPPAT: PIN;          %STEP A6.b
52          }}
53  }}
54
55  %%%% TreC APP %%%%
56  entity Service_Provider(Actor, OTPPAT, ADC, Patient: agent, Ch_T2O, Ch_O2T,
57  Ch_P2T, Ch_T2P: channel, Request: text) {
58
59  body{%of SP
60  select{on(Patient -Ch_P2T-> Actor: Request):{  %STEP A1
61  Actor -Ch_T2O-> OTPPAT: Actor;  %STEP A2
62          select{on(OTPPAT-Ch_O2T->Actor:{ADC.?Patient.Actor}_inv(pk(ADC))):{%A8
63          SP_authn_U_on_Request:(Request) := Request;
64          }}
65    }}
66  }}
67
68  %%%% OTP-PAT APP %%%%
69  entity User_Agent(Actor, TreC, ADC, Patient: agent, Ch_T2O, Ch_O2T, Ch_O2A,
70  Ch_A2O, Ch_P2O, Ch_O2P: channel, Seed: seed, Token_IDP: token, CTime: time) {
71
72  symbols
73  KeyHash: key_hash;
74  Metadata: text;
75  PINRequest: text;
76  PIN: pin;
77
78  body{%of UA
79  iknows(CTime);
80
81  select {on(TreC -Ch_T2O-> Actor: TreC & %STEP A2
82  infoMobile -> contains((TreC,?KeyHash))):{ %STEP A3
83  Actor -Ch_O2A-> ADC: TreC.KeyHash.Token_IDP; %STEP A4
84    select {on(ADC -Ch_A2O-> Actor: ?Metadata ):{ %STEP A5.a
85    userconsent; %STEP A5.b
86    Actor -Ch_O2P-> Patient: PINRequest; %STEP A6.a
87          select{on(Patient -Ch_P2O-> Actor: ?PIN &
```

```
88          pinIDOTP(Actor) -> contains(?PIN)):{ %STEP A6.b
89          Actor -Ch_O2A-> ADC: IDP_authn_UA_on_OTP:(otp_generation(Seed, CTime)).
90          TreC.KeyHash.Token_IDP; %STEP A7.c - A6.d
91          select{on(ADC-Ch_A2O-> Actor:{ADC.?Patient.TreC}_inv(pk(ADC))):{%STEP A7
92          Actor -Ch_O2T-> TreC: {ADC.Patient.TreC}_inv(pk(ADC)); %STEP A8
93          }}
94          }}
95   }}
96  }}
97 }}
98
99
100 %%%% ADC %%%%
101 entity Identity_Provider(Actor, OTPPAT, TreC: agent, Ch_O2A, Ch_A2O: channel,
102 CTime: time){
103
104 symbols
105 KeyHash: key_hash;
106 Metadata: text;
107 Token_IDP: token;
108 Seed: seed;
109 Patient: agent;
110
111 body{%of ADC
112  select{ on(OTPPAT -Ch_O2A-> Actor: ?TreC.?KeyHash.?Token_IDP & %STEP A4
113  metadataDB(Actor) -> contains((?KeyHash, ?Metadata)) ):{
114  Actor -Ch_A2O-> OTPPAT: Metadata; %STEP A5.a
115   select{ on(OTPPAT -Ch_O2A-> Actor: IDP_authn_UA_on_OTP:(otp_generation(?Seed,
116   CTime)).?TreC.?KeyHash.?Token_IDP & %STEP A6.d
117   trustedSPs(Actor) -> contains((?TreC,?KeyHash))):{
118     select{ on(usersDB(Actor)-> contains((?Patient, Seed, Token_IDP))):{
119     Actor -Ch_A2O-> OTPPAT: {Actor.Patient.TreC}_inv(pk(Actor)); %STEP A7
120    }}
121   }}
122  }}
123 }}
124
125 body {%session
```

```
126 CTime:=fresh();
127
128 %%channel properties
129 link(Ch_T2O,Ch_O2T); %CA1
130 authentic_on(Ch_T2O, TreC); %CA2
131 unilateral_conf_auth(Ch_O2A, Ch_A2O, ADC); %CA3
132 confidential_to(Ch_P2O, OTPPAT); %UBA1
133 authentic_on(Ch_P2O, Patient); %UBA2
134
135 %%a new session is built
136 new User(Patient, OTPPAT, TreC, Ch_P2O, Ch_O2P,Ch_P2T, Ch_T2P, PIN, Request);
137 new Service_Provider(TreC, OTPPAT, ADC, Patient, Ch_T2O, Ch_O2T, Ch_P2T,
138 Ch_T2P, Request);
139 new User_Agent(OTPPAT, TreC, ADC, Patient, Ch_T2O, Ch_O2T, Ch_O2A, Ch_A2O,
140 Ch_P2O, Ch_O2P, Seed, Token_IDP, CTime);
141 new Identity_Provider(ADC, OTPPAT, TreC, Ch_O2A, Ch_A2O, CTime);
142 }
143
144 goals
145 SP_authn_U_on_Request:(_) Patient*->>TreC; %goal (G1)
146 IDP_authn_TP_on_OTP:(_) OTPPAT*->>ADC;  %goal (G2)
147
148 } %end of Session
149
150 body { %of environment
151
152 %iknows(token_idp); %uncomment to remove BA1 and AA
153 %iknows({|seedotp|}_pinUser); %uncomment to remove BA1 and AA
154 %iknows(pinUser); %uncomment to remove BA2 and AA
155
156 trustedSPs(adc)->add((trec,keyhash));
157 trustedSPs(adc)->add((i,keyhash_i));
158 infoMobile->add((trec,keyhash));
159 infoMobile->add((i,keyhash_i));
160 pinIDOTP(optpat)->add(pinUser);
161 metadataDB(adc) ->add((keyhash,metadata));
162 metadataDB(adc) ->add((keyhash_i,metadata_i));
163 usersDB(adc)-> add((patient, seedotp, token_idp));
```

```
164
165 %Sessions
166 new Session(trec, optpat, adc, patient, ch_t2o, ch_o2t, ch_o2a, ch_a2o,
167 ch_p2o, ch_o2p, ch_p2t, ch_t2p, seedotp, token_idp, ctime, pinUser, request1);
168 new Session(trec, optpat, adc, patient, ch_t2o1, ch_o2t1, ch_o2a1, ch_a2o1,
169 ch_p2o1, ch_o2p1, ch_p2t1, ch_t2p1, seedotp, token_idp,ctime,pinUser,request2);
170 %new Session(trec, optpat, i, patient, ch_t2o1, ch_o2t1, ch_o2i, ch_i2o,
171 %ch_p2o1, ch_o2p1, ch_p2t1, ch_t2p1, seedotp, token_idp, ctime, pinUser,
172 % request2);  uncomment to remove TA
173 }
174
175 } %end of Environment
```

B.2 DigiMat-Lab ASLan++

We report here the complete ASLan++ specification for the DigiMat-Lab scenario.

```
1  specification cie
2  channel_model ACM
3
4  entity Environment{
5
6  types
7  pin < text;
8  cookie < text;
9  requestu < text;
10 opid < text;
11
12 symbols
13 citizen, cie, car, browser, ipzs, idp: agent;
14 nonpublic session_cookie: cookie;
15 nonpublic pin1: pin;
16 nonpublic pin2: pin;
17 request1: requestu;
18 request2: requestu;
19 nonpublic usersDB(agent): agent*cookie set;
```

```
nonpublic opDB(agent): opid*cookie set;
nonpublic pin_CIE(agent): pin*pin set;

ch_u2c, ch_ipzs2u, ch_u2ipzs, ch_c2b, ch_b2c, ch_b2idp, ch_idp2b, ch_b2ipzs,
 ch_ipzs2b, ch_ipzs2cie, ch_cie2ipzs: channel;
ch_u2c1, ch_ipzs2u1, ch_u2ipzs1, ch_c2b1, ch_b2c1, ch_b2idp1, ch_idp2b1,
 ch_b2ipzs1, ch_ipzs2b1, ch_ipzs2cie1, ch_cie2ipzs1: channel;
ch_b2i, ch_i2b:channel;


%%%%%%% SESSION %%%%%%%
entity Session(Citizen, CIE, Car, Browser, IPZS, IdP:agent, Ch_U2C, Ch_IPZS2U,
Ch_U2IPZS, Ch_C2B, Ch_B2C, Ch_B2IDP, Ch_IDP2B, Ch_B2IPZS,Ch_IPZS2B, Ch_IPZS2CIE,
Ch_CIE2IPZS:channel, Request:requestu, PIN1: pin, Cookie:cookie,PIN2: pin){


%%%% Citizen %%%%
entity User(Actor, Car, IPZS:agent, Ch_U2C, Ch_IPZS2U, Ch_U2IPZS:channel,
 Request:requestu, PIN1:pin){

symbols
OpID: opid;

body{%of user
       Actor -Ch_U2C-> Car: SP_authn_U_on_Request:(Request); %STEP A1
       select{on(IPZS -Ch_IPZS2U-> Actor: ?OpID):{ %STEP A4.c
       Actor -Ch_U2IPZS-> IPZS: PIN1; %STEP A4.d
       }}
}}


%%%% Car APP %%%%
entity Service_Provider(Actor, Citizen, Browser, IdP:agent, Ch_U2C, Ch_C2B,
 Ch_B2C: channel, Request: requestu){

body{%of SP
  select{on(Citizen -Ch_U2C-> Actor: Request):{ %STEP A1
  Actor -Ch_C2B-> Browser: Actor; %STEP A2
```

```
58    select{on(Browser -Ch_B2C-> Actor: {IdP.?Citizen.Actor}_inv(pk(IdP))):{%A7
59    SP_authn_U_on_Request:(Request) := Request;
60    }}
61   }}
62 }}
63
64
65 %%% Browser %%%%
66 entity User_Agent(Actor, Car, IdP, IPZS, CIE:agent, Ch_C2B, Ch_B2IDP, Ch_IDP2B,
67  Ch_B2IPZS, Ch_IPZS2B, Ch_B2C:channel, Cookie:cookie){
68
69 symbols
70 Citizen:agent;
71 OpID:opid;
72
73 body{%of UA
74 select{on(Car -Ch_C2B-> Actor: Car):{ %STEP A2
75 Actor -Ch_B2IDP-> IdP: Car.Cookie; %STEP A3
76 select{on(IdP -Ch_IDP2B-> Actor: ?OpID.Car.IdP):{ %STEP A4.a
77 Actor -Ch_B2IPZS-> IPZS: OpID.Car.IdP; %STEP A4.b
78 select{on(IPZS -Ch_IPZS2B-> Actor: {OpID.Car.IdP}_inv(pk(CIE))):{ %STEP A4.g
79 Actor -Ch_B2IDP-> IdP: {OpID.Car.IdP}_inv(pk(CIE)).Cookie; %STEP A4.h
80 select{on(IdP -Ch_IDP2B-> Actor: {IdP.?Citizen.Car}_inv(pk(IdP))):{ %STEP A6
81 Actor -Ch_B2C-> Car: {IdP.Citizen.Car}_inv(pk(IdP)); %STEP A7
82 }} }} }} }}
83 }}
84
85 %% IPZS App %%%%
86 entity Token_Provider(Actor, Browser, Citizen, CIE, IdP:agent, Ch_B2IPZS,
87  Ch_IPZS2U, Ch_U2IPZS, Ch_IPZS2CIE, Ch_CIE2IPZS, Ch_IPZS2B:channel, PIN2:pin){
88
89 symbols
90 Car: agent;
91 PIN1: pin;
92 OpID: opid;
93
94 body{%of TP
95 select{ on(Browser -Ch_B2IPZS-> Actor: ?OpID.?Car.IdP):{ %STEP A4.b
```

```
96  Actor -Ch_IPZS2U-> Citizen: OpID; %STEP A4.c
97  select{ on(Citizen -Ch_U2IPZS-> Actor: ?PIN1):{ %STEP A4.d
98  Actor -Ch_IPZS2CIE-> CIE: OpID.Car.IdP.PIN1.PIN2; %STEP A4.e
99  select{on(CIE -Ch_CIE2IPZS-> Actor: {OpID.Car.IdP}_inv(pk(CIE))):{ %STEP A4.f
100 Actor -Ch_IPZS2B-> Browser: {OpID.Car.IdP}_inv(pk(CIE)); %STEP A4.g
101 }} }} }}
102 }}
103
104
105 %% CIE %%%%
106 entity Token(Actor, IPZS, IdP:agent, Ch_IPZS2CIE, Ch_CIE2IPZS:channel, PIN1,
107  PIN2: pin){
108
109 symbols
110 Car:agent;
111 OpID: opid;
112
113 body{%of Token
114   select{on(IPZS -Ch_IPZS2CIE-> Actor: ?OpID.?Car.IdP.PIN1.PIN2):{ %STEP A4.e
115    Actor -Ch_CIE2IPZS-> IPZS:{OpID.Car.IdP}_inv(pk(Actor));%STEP A4.f
116    %IDP_authn_TP_on_OTP:({OpID.Car.IdP}_inv(pk(Actor)));
117   }}
118 }}
119
120
121 %% IdP_IPZS %%%%
122 entity Identity_Provider(Actor, Browser, Citizen, CIE: agent, Ch_B2IDP,
123  Ch_IDP2B:channel){
124
125 symbols
126 Car: agent;
127 Cookie: cookie;
128 OpID: opid;
129
130 body{%of IDP
131 select{on(Browser -Ch_B2IDP-> Actor: ?Car.?Cookie):{ %STEP A3
132 OpID:=fresh();
133 opDB(Actor)-> add((OpID,Cookie));
```

```
Actor -Ch_IDP2B-> Browser: OpID.Car.Actor; %STEP A4.a
select{on(Browser-Ch_B2IDP->Actor:{OpID.Car.Actor}_inv(pk(CIE)).?Cookie& %A4.h
%IDP_authn_TP_on_OTP:({OpID.Car.Actor}_inv(pk(CIE))).?Cookie &
usersDB(Actor) -> contains((Citizen, ?Cookie)) &
opDB(Actor)-> contains((OpID,?Cookie))):{
Actor -Ch_IDP2B-> Browser: {Actor.Citizen.Car}_inv(pk(Actor)); %STEP A6
  }} }}
}}

body {%session

%%channel properties
confidential_to(Ch_B2C, Car);                %CA1
unilateral_conf_auth(Ch_B2IDP, Ch_IDP2B, IdP); %CA2
confidential_to(Ch_IPZS2B, Browser);                %CA3
unilateral_conf_auth(Ch_IPZS2CIE, Ch_CIE2IPZS, CIE); %CA4
confidential_to(Ch_U2IPZS, IPZS);    %UBA1
authentic_on(Ch_U2IPZS, Citizen);    %UBA2
authentic_on(Ch_IPZS2CIE,IPZS); %UBA3

%%a new session is built
new User(Citizen, Car, IPZS, Ch_U2C, Ch_IPZS2U, Ch_U2IPZS, Request, PIN1);
new Service_Provider(Car, Citizen, Browser, IdP, Ch_U2C, Ch_C2B, Ch_B2C,
 Request);
new User_Agent(Browser, Car, IdP, IPZS, CIE, Ch_C2B, Ch_B2IDP, Ch_IDP2B,
 Ch_B2IPZS, Ch_IPZS2B, Ch_B2C, Cookie);
new Token_Provider(IPZS, Browser, Citizen, CIE, IdP, Ch_B2IPZS, Ch_IPZS2U,
 Ch_U2IPZS, Ch_IPZS2CIE, Ch_CIE2IPZS, Ch_IPZS2B, PIN2);
new Token(CIE, IPZS, IdP, Ch_IPZS2CIE, Ch_CIE2IPZS, PIN1, PIN2);
new Identity_Provider(IdP, Browser, Citizen, CIE, Ch_B2IDP, Ch_IDP2B);
}

goals
SP_authn_U_on_Request:(_) Citizen*->>Car; %goal (G1)
%IDP_authn_TP_on_OTP:(_) CIE*->>IdP;  %goal (G2)
} %end of Session

body { %of environment
```

```
172
173 %iknows(session_cookie);      %uncomment to remove BA1 and AA
174 %iknows(pin1);             %uncomment to remove BA2 and AA
175 %iknows(pin2);             %uncomment to remove BA1 and AA
176
177 usersDB(idp)-> add((citizen, session_cookie));
178
179 %Sessions
180 new Session(citizen, cie, car, browser, ipzs, idp, ch_u2c, ch_ipzs2u,
181  ch_u2ipzs, ch_c2b, ch_b2c, ch_b2idp, ch_idp2b, ch_b2ipzs, ch_ipzs2b,
182   ch_ipzs2cie, ch_cie2ipzs, request1, pin1, session_cookie, pin2);
183 new Session(citizen, cie, car, browser, ipzs, idp,ch_u2c1, ch_ipzs2u1,
184  ch_u2ipzs1, ch_c2b1, ch_b2c1, ch_b2idp1, ch_idp2b1, ch_b2ipzs1, ch_ipzs2b1,
185   ch_ipzs2cie1, ch_cie2ipzs1, request2, pin1, session_cookie, pin2);
186 %new Session(citizen, cie, car, browser, ipzs, i,ch_u2c1, ch_ipzs2u1,
187 % ch_u2ipzs1, ch_c2b1, ch_b2c1, ch_b2i, ch_i2b, ch_b2ipzs1, ch_ipzs2b1,
188 %  ch_ipzs2cie1, ch_cie2ipzs1, request2, pin1, session_cookie, pin2, opid2);
189 }
190
191 } %end of Environment
```