A Privacy-Enhanced OAuth 2.0 based Protocol for Smart City Mobile Applications

Victor Sucasas, Georgios Mantas, Saud Althunibat, Leonardo Oliveira,
Angelos Antonopoulos, Ifiok Otung and Jonathan Rodriguez

***Abstract*—In the forthcoming Smart City scenario, Service Providers will require users to authenticate themselves and authorize their mobile applications to access their remote accounts. In this scenario, OAuth 2.0 has been widely adopted as a de facto authentication and authorization protocol. However, the current OAuth 2.0 protocol specification does not consider the user privacy issue and presents several vulnerabilities that can jeopardize users' privacy rights. Therefore, in this paper we propose an OAuth 2.0 based protocol for Smart City mobile applications that addresses the user privacy issue by integrating a pseudonym-based signature scheme and a signature delegation scheme into the OAuth 2.0 protocol flow. The proposed solution allows users to self-generate user-specific and app-specific pseudonyms on-demand and ensure privacy-enhanced user authentication at the Service Provider side. The proposed protocol has been validated with Proverif and its performance has been evaluated in terms of time and space complexity. Results show that the proposed protocol can provide users with efficient and effective means to authenticate towards service providers while preventing user tracking and impersonation from malicious entities located in the network side or in the users' mobile device.**[1]

I. Introduction

Over the past six decades, the urban population of the world has grown rapidly and it is expected to grow by 2.5 billion urban dwellers by 2050 [1]. Hence, due to the projected explosion of urban population in the coming decades, cities worldwide will face major sustainable development challenges in terms of economic development, social development and environmental protection. Consequently, there is an urgent need for cities around the world to become sustainable, under such conditions, in order to address these challenges [1].

Towards this direction, the emerging concept of Smart City is considered as a promising solution to achieve sustainable cities providing avenues for economic growth, technological progress and prosperity [2], [3], [4], [5]. As the International Telecommunication Union (Telecommunication Standardization sectior, ITU-T) Focus Group on Smart Sustainable Cities defined in [2], a Smart Sustainable City is an innovative city that makes use of Information and Communication Technologies (ICTs) and other means to enhance the quality of life of its citizens, improve efficiency of urban operation and services, and increase competitiveness. At the same time, the Smart Sustainable City ensures that it satisfies the needs of present and future generations with respect to economic, social and environmental aspects [2]. Particularly, in future Smart Cities, ICTs are expected to act as the key medium that will address all the potential urban challenges to improve the quality of life of its citizens.

In this context, Smart City mobile applications running on citizens' smartphones can play an important role in the improvement of citizens' quality of life since they can enable citizens to access a plethora of personalized services (e.g., m-health services, transportation services) without limitations on time and location. Smart City mobile applications can be provided not only by the Service Providers (SPs) that are responsible for developing citizens' personalized services but also by third-party developers that leverage the Application Programming Interfaces (APIs) of the SPs [6]. Nevertheless, personalized services may handle personal and sensitive information such as citizens' profile data, contextual data or even vital data and thus, secure access to them is a vital concern. Therefore, the appropriate authentication and authorization mechanisms that allow the delegated access to personalized services are required.

Today, the OAuth protocol, whose latest version is OAuth 2.0 [7], is the most widely used protocol for implementing authentication and authorization functionality in mobile applications [8],[9]. However, although OAuth security for web applications has been studied sufficiently, its secure usage for mobile applications needs to be further investigated due to the fact that OAuth was initially designed to enable end-users to authorize third-party websites to access their resources stored on a remote Resource Server (e.g., SP) on their own behalf. Moreover, it is worthwhile to mention that according to [9], [10], and [11], OAuth 2.0 has already been vulnerable to mobile malware attacks targeting end-user's credentials. By exploiting mobile malware, or by eavesdropping the message flow during the OAuth 2.0 protocol execution, attackers are capable of retrieving sensitive information to link the citizen's real identity to his/her mobile applications [12], [13], [14], [15]. The realease of such information enables unauthorized entities to profile users' activities. Consequently, privacy

[1]V. Sucasas, G. Mantas, L. Oliveira and J. Rodriguez are with the Instituto de Telecomunicações, Aveiro, Portugal (email: vsucasas@av.it.pt, gimantas@av.it.pt, leonardooliveira@ua.pt, jonathan@av.it.pt).

S. Althunibat is with Al-Hussein Bin Talal University, Ma'an, Jordan (email: saud.althunibat@ahu.edu.jo).

I. Otung is with Mobile and Satellite Communications Research Group, School of Engineering, with University of South Wales, (email: ifiok.otung@southwales.ac.uk).

A. Antonopoulos is with Telecommunications Technological Center of Catalonia (CTTC/CERCA), Barcelona, Spain (email: angelos.antonopoulos@cttc.es)

concerns arise for Smart City mobile applications enabling citizens to access their personalized services [16], [17].

Therefore, in this paper, we extend our work in [18] and propose a privacy-enhanced OAuth 2.0 based protocol for Smart City mobile applications so that they can provide access to citizen's personalized services without the leakage of private information to unauthorized entities. In particular the proposed protocol ensures privacy of the user identity and the activity of the different mobile apps running on the user's mobile device. The proposed privacy-enhanced OAuth 2.0 based protocol integrates two signature schemes into the Step 2 of OAuth 2.0 protocol flow with authorization code grant type. Namely, the Step 2 corresponds with the user authentication process, and the integrated signature schemes consist of: i) a pseudonym-based signature scheme; and ii) a signature delegation scheme. In particular, we modified our previous proposed protocol in [18] by correcting a vulnerability affecting the users' pseudonyms unlinkability. In our previous proposed protocol an attacker was able to link different pseudonyms from the same user by performing some bilinear pairing operations using the user's pseudonyms and public parameters. We have validated the proposed protocol with applied pi calculus (Proverif [19]) in terms of observational equivalence and unreachability properties, which proves the unlinkability properties of the users' pseudonyms and the protocol's resilience against credential thief. It is worth commenting that the same tests were used to identify the vulnerability affecting our previous work, which failed in terms of observational equivalence. Furthermore, we implemented the integrated signature schemes and evaluated the time complexity of the signature verification process in the user side. Finally, we have also evaluated the time and space complexity of the protocol in the Privacy Server side.

Following the introduction, this paper is organized as follows: Section II presents related work on OAuth 2.0 for mobile apps and discusses OAuth 2.0 security issues for mobile apps. In Section III, we provide the system model where the proposed protocol is applied. Section IV shows the threat model, from which the system requirements are derived. The system requirements are defined in section V. Furthermore, the proposed privacy-enhanced OAuth 2.0 based protocol is presented in Section VI. In Section VII the security analysis is provided. Section VIII includes the performance evaluation. Finally, Section IX concludes the paper.

II. Related Work

A. OAuth 2.0 and Mobile Apps

The OAuth 2.0 protocol, as described in the RFC 6749 [7] (evolved from the previous specification of OAuth1 [20]), defines the interactions among the user (*resource owner* in the protocol specification), the mobile application (*client* in the protocol specification), back-end server of the service provider where the user's resources are hosted (*resource server* in the protocol specification), and the authorization server that is the entity granting the mobile application access to the user's resources hosted in the back-end server of the service provider on behalf of the user.

According to the OAuth 2.0 protocol, the Client requests access to the Resource Owner's protected resources, hosted by the Resource Server, and is issued a different set of credentials than those of the Resource Owner. In particular, the Client is issued an access token (i.e., credentials) by the Authorization Server with the approval (i.e., authorization) of the Resource Owner. The access token is a string denoting a specific scope, lifetime, and other access attributes, and is used by the Client to access the Resource Owner's protected resources. The authorization of the Resource Owner is expressed in the form of an authorization grant that the Client uses to request the access token from the Authorization Server. OAuth 2.0 provides four different methods so that the Client can obtain the access token. These methods are referred to as grant types and are the following [7]: authorization code, implicit, resource owner password credentials, and client credentials. However, as it is also claimed in [9], the authorization code grant type and the implicit grant type are the two grant types that have been widely used in practice for native applications (i.e., desktop applications, mobile apps). Besides, both these grant types are redirection-based flow, which means that the Client should be capable of interacting with the Resource Owner's user-agent (i.e., web browser) and capable of receiving incoming request, via redirection, from the Authorization Server. In this paper, our focus is on the OAuth 2.0 protocol flow with authorization code grant since this grant type supports Client authentication which is essential to increase the security level of Smart City mobile apps that can have access to sensitive information. However, as it is also recommended for native apps in RFC 6749 [7], we consider the use of the authorization code grant type without using Client credentials, because of the fact that mobile applications are not able to keep users' credentials confidential.

According to the implementations of the OAuth 2.0 protocol flow with authorization code grant for mobile apps, the client/mobile application requests access to the user's account/resources on behalf of the user. For that, the mobile app deploys an user-agent to interact with the user. The user-agent can be the system native browser or an embedded web browser, which is a User Interface (UI) view component integrated in the mobile app in order to render online content within the mobile app. In terms of functionality, the native web browser and the embedded web browser are similar in the context of the OAuth 2.0 protocol flow. Hence, they both are referred simply as browser [9], [10].

The abstract OAuth 2.0 protocol flow with authorization code grant for mobile apps is presented in Fig. 1. The mobile app initiates the flow by redirecting the User-Agent (i.e., browser) to the Authorization Server in order to request the authorization code (Step 1). Afterwards, the Authorization Server authenticates the user, via the browser, and requests the user input to grant or deny the authorization to the mobile application (Step 2). If the user accepts the mobile app's request the Authorization Server redirects the browser back to the mobile app including

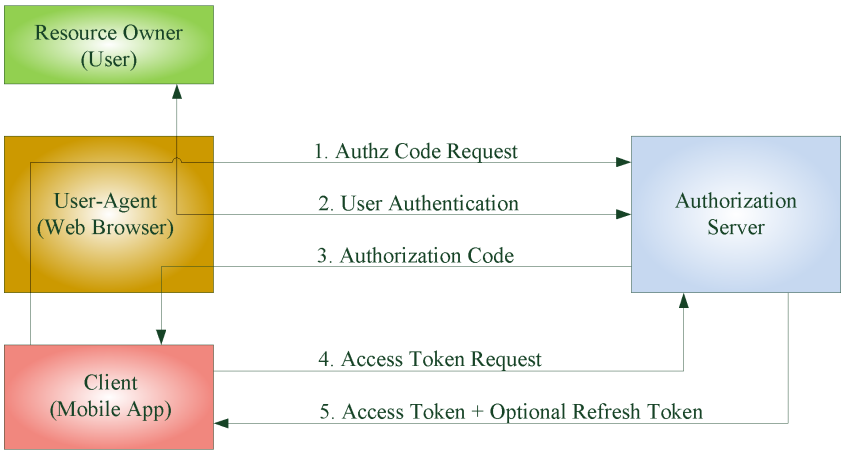

Fig. 1. Abstract OAuth 2.0 Protocol Flow with Authorization Code Grant type for Mobile Apps.

the authorization code (Step 3). With the authorization code, the mobile app can request an access token from the Authorization Server that can be used to access the user's resources hosted in the resource server (Step 4). Finally, if all validations are completed successfully by the Authorization Server, then an access token and, optionally, a refresh token are issued to the mobile app (Step 5) [9], [10]. Detailed description of this protocol flow is provided in [7] and [18]. Our proposed protocol is based on the OAuth 2.0 protocol with authorization code grant and improves the user privacy by integrating two cryptographic schemes into the user authentication process in Step 2. Therefore, our focus is on the design of the two cryptographic schemes and their seamless integration in Step 2.

B. OAuth 2.0 Security for Mobile Apps

During the past few years, the security community has put a lot of effort on OAuth security for web applications [21], [22], [23]. However, to the authors' best knowledge, very few works are available in the literature that are explicitly focused on OAuth security for mobile applications. In the following, we present the main contributions to this the area.

According to [9], although the OAuth protocol was initially designed for website authorization, nowadays, it has been adopted as a de-facto standard for authentication/authorization in mobile applications. However, as it is shown in [9], several concepts of the OAuth protocol as well as a number of its key steps are not clear or specified explicitly when they are considered for mobile platforms, and as a result, many developer misconceptions about OAuth implementations in mobile environments arise. Therefore, in [9], the authors conduct a thorough evaluation of the OAuth protocol in the context of mobile applications, so developers can avoid misconceptions leading to mistakes that cause vulnerabilities related to OAuth implementation for mobile platforms. Based on the authors' study, these mistakes are caused by the fact that the mobile app developers: a) store the application's secret locally inside the application, b) treat authorization and authentication as the same problem, c) use arbitrary mechanisms to transmit secret tokens, and d) modify the existing OAuth 2.0 protocol flow. Among the lessons and conclusions highlighted by the authors, it is worth mentioning the difficulty for service providers to verify that access tokens are transmitted to the intended recipients, which requires to tag users with globally unique identifiers. Also, the interaction with the user through a UI, asking for his/her acceptance in the authorization process and informing him/her about the scope of such authorization, should never be avoided.

Finally, the last lesson is related to the fact that when the OAuth protocol is used for authentication, the user's mobile device must be considered as untrusted as well as the communication channel. This consideration has two implications: a) the mobile application should not perform parts of security-related protocols, since it could fake the results of such protocols, or leak sensitive information; and b) it must be assumed that the attacker can eavesdrop or modify any data transmitted from the mobile app [15]. Indeed, in [10], the authors present a practical set up in which a malicious app installed on the user's mobile device can steal the user's credentials or the access token. The proposed solution in [10] is a trusted app, called OAuth Manager, in which the critical OAuth components are implemented. In our proposed approach we also use the approach of providing a trusted app, called Privacy app (P-app), that integrates a secure UI component to obtain the user's input safely and performs the cryptographic operations required for the proposed pseudonym-based signature schemes.

C. Pseudonym-based Systems

A pseudonym-based signature scheme can provide privacy-preserving authentication, since pseudonyms can serve as identifiers for entities while still preserving the entities' anonymous state (i.e., a state in which the users cannot be distinguished from other entities in the set of users [24]). Entities can hold many pseudonyms representing the entity, its roles or functions, or the different relations of the entity with different organizations [25]. Different pseudonyms of the same entity are not linkable between each other and do not leak any information about the entity's real identity. However, a Certification

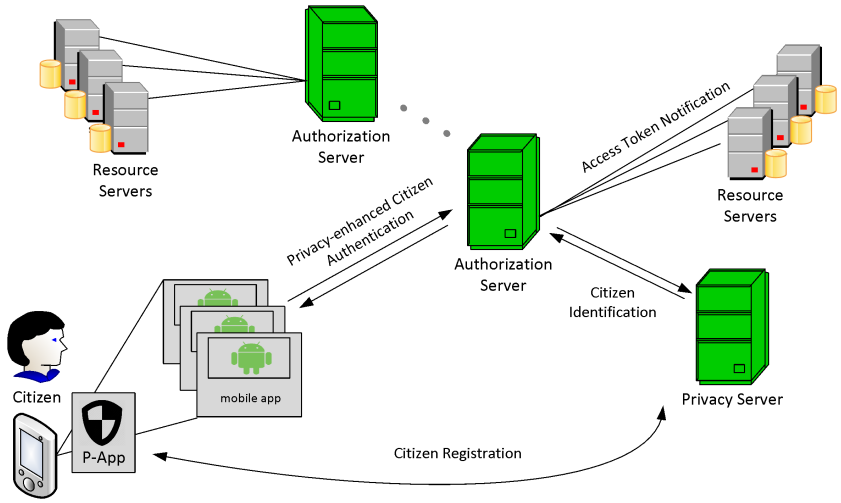

Fig. 2. System model of the proposed privacy-enhanced OAuth 2.0 based protocol for Smart City mobile apps.

Authority (CA) can be enabled to link a pseudonym to the real identity of the pseudonym's owner, which is referred as anonymity revocation and usually proposed as a mechanism to counteract against misusers that user their anonymity state fraudulently [26]. It is worth mentioning that in our proposed system, revocation is used to enable user controlled identification.

The are different crypto-systems that enable the implementation of pseudonyms such as digital anonymous credentials, where users can generate pseudonyms to represent themselves towards different organizations, and prove the possession of a credential that validates such pseudonyms. According to [27], the same credential can be transferred to validate pseudonyms in multiple organizations. The application of a PKI might seem simpler than previous solutions in some scenarios, such as in Vehicular Networks (VANETs). Indeed, a number of research works have suggested that users can sign messages with private keys, which can be verified with the corresponding public keys representing their pseudo-identities [28]. However, this requires a CA to generate, distribute and periodically renew a large number of public key certificates which have to be also transmitted together with the signed messages. A simpler solution is Identity Based Cryptography (IBC) [29] but using pseudo-identities instead of the real identity of the users [30]. With IBC users use the pseudonyms as public keys for which the CA can generate private keys and sign messages with these pseudonyms. Revocation is still possible when the CA issues the private keys [25],[31].

Users can also be allowed to self-generate their pseudonyms, however, the pseudonym self-generation feature must go together with an efficient manner to control the users' generation capabilities and to enable the revocation. A possibility to control the pseudonym self-generation is to require the pseudonyms to be signed by a trusted third party [32][25]. However, the requirement for the interaction with a trusted third party limits the scalability and the efficiency of the system. Group signatures or ring signatures can also be used to address the scalability issue [33],[34], since the users, belonging to a group, can generate their own pseudonyms and sign these pseudonyms with the group signing key. However, this approach has already been discarded in other scenarios such as VANETs, due to the complexity, since it requires users to self-issue a certificate on the pseudonyms and transmit such certificates together with the pseudonym and the signed message.

To be considered efficient, a pseudonym-based signature scheme should enable self-generation of pseudonyms with a single credential from the CA, hence limiting the interactions between the users and the CA. Anonymity revocation by the CA should still be feasible and efficient when pseudonyms are generated in the user side. Also, the system should provide all-or-nothing transferability [26] to discourage users from generating pseudonyms and sharing these pseudonyms with other users.

The work in [35] provides two signature schemes that are close to provide these features. In the second scheme the authors propose a signature scheme with time-controlled pseudonym variation, which limits a pseudonym self-generation to one pseudonym per timeslot. The time-controlled pseudonym variation also allows to pre-compute the users' pseudonyms in the CA [36] which enables a fast revocation mechanism. However, the controlled self-generated pseudonyms are not unlinkable. It is possible to link the pseudonyms generated with the same credential in different time-slots. The proposed protocol for privacy-preservation in OAuth 2.0 uses an adaptation of the latter signature scheme (presented in [35] and adapted in [36]) in the Authorization Server side, while the user-side uses a signature delegation scheme that is not affected by pseudonym linkability attacks and that can be efficiently combined with the pseudonym-based signature scheme [35].

III. System Model

The proposed system model architecture is depicted in Figure 2 and composed by the following entities: i) The Citizen - also referred as user - which is the entity subscribed to one or several remote services and owning an account in one or several resource servers; ii) Mobile App -

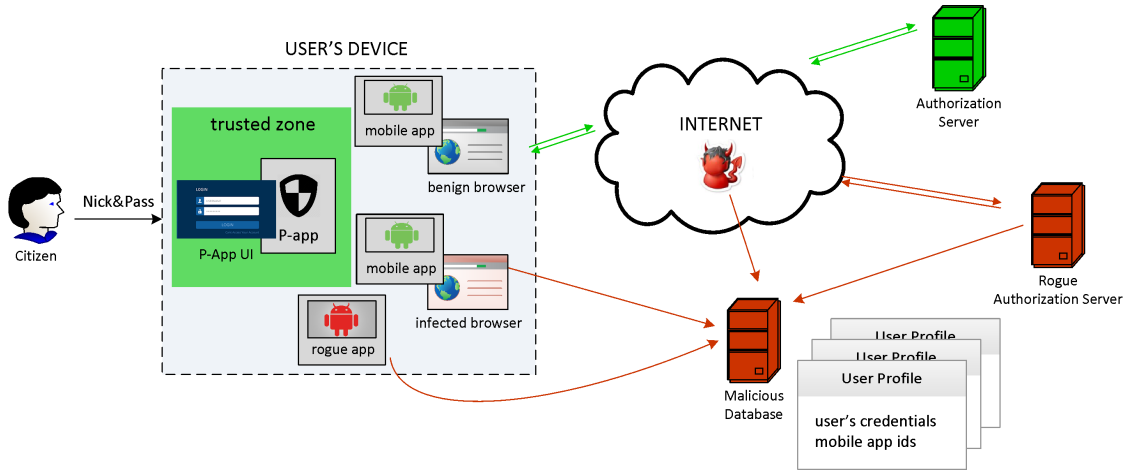

Fig. 3. Threat model consisting of an attacker extracting information to make a database with users' activity profiles.

referred to as client in the OAuth 2.0 specification - which is a mobile application installed in the users' mobile devices that provides access to the users' subscribed services, and that can access users' remote resources when authorized by the user; iii) Privacy App (P-App), which is a mobile application installed in the users' mobile device and that assists in the performance of the proposed privacy-preserving protocol by performing the cryptographic operations. The P-App also stores the non-sensitive information required for the performance of the proposed protocol; iv) Resource servers (RSs), which are the servers hosting the users' accounts and providing personalized services to citizens; v) Authorization Servers (ASs) the servers providing access control mechanisms to enable users to authenticate and authorize mobile applications to access on their behalf to remote users' resources; vi) Privacy Server (PS), which enables privacy-preserving user authentication by storing users' real identities and triggering the anonymity revocation process when authorized by the user. The PS provides the cryptographic mechanisms for the proposed privacy-enhanced OAuth 2.0 based protocol.

IV. Threat Model

The threat model, as depicted in Fig. 3, considers an attacker whose motivation is to profile: a) the user's credentials and b) the id of the mobile app running on the smartphone and using OAuth 2.0 protocol to obtain authorized access to user's resources on behalf of him/her. The main objective of the attacker is to create the user's profile that allow the attacker to infer the user's identity from the user's credentials and link the user's identity to the mobile apps running on the user's device. To achieve this, the attacker should be present in the user's device, in the network (i.e., Internet), or in the AS side in order to extract the appropriate information (i.e., user's credentials and mobile app ids) during the execution of the user authentication phase of the OAuth 2.0 protocol.

The model assumes the following vulnerabilities in the user's mobile device due to the installation of malware [10]:

- A rogue mobile app can extract the information that the user inputs in the mobile app embedded interface.
- A trustful mobile application may use a compromised native web browser, which can extract the information typed by the user, e.g. the user's credentials.
- The attacker can inspect the source code of the P-app installed in the user's mobile device and obtain the stored data. However, the user interface of the P-app and the cryptographic operations performed by the P-app during the OAuth 2.0 protocol execution cannot compromised.

The model also assumes the following vulnerabilities in the network side and the Authroization Server side as described in the OAuth 2.0 threat model in RFC6819 [37]:

- The attacker has full access to the network between the client and the Authorization Server. The attacker may eavesdrop on any communications between those parties.
- The attacker can impersonate an Authorization Server and perform the OAuth 2.0 protocol with the user.

The threat model assumes that the attacker is provided with effective means to infer the user's identity from the information transmitted through the OAuth 2.0 message flow, i.e. the user's credentials. The attacker can also link the activity of different mobile apps to a single user by linking different OAuth 2.0 protocol executions. It is worth mentioning that the threat model covers only the application layer, hence the attacker does not perform any traffic analysis using the IP or MAC addresses in the communications intercepted. Note that such an attack can be avoided by applying solutions at the network layer like TOR [38].

A. Attack example and proposed solution

In this section we provide an attack scenario consisting of four users and the attacker who extracts different information about the users based on his/her capabilities as seen in Fig. 4. In this scenario the attacker knows that

in the scenario there are four users and their identities, but the attacker cannot not observe physically their activities in their mobile devices. The attacker is given full access to the information extracted during the OAuth 2.0 protocol execution of every mobile application running on the users' mobile devices. At a given moment, one of the users (Jasper) activates three mobile apps whereas another user (Jimmy) activates one mobile app. The other two users remain inactive. The attacker tries to obtain the information about who is running which application.

In case A, Fig. 4, the attacker extracts the users' credentials and app ids then the attacker is able to profile the two active users (Jasper and Jimmy) and learns which mobile apps are being using by each user. In the second case (case B), the OAuth 2.0 protocol is protected using the proposed privacy-preserving mechanism. However, the attacker is given the advantage to link the pseudonyms belonging to the same user. In this case, the attacker does not profile the specific mobile apps to specific users, but he/she learns that a single user is using the mobile apps with app_id 1, 2, and 4, and another user is using the mobile app with app_id 3. The attacker also knows that the other two users are inactive. This information is not sufficient to profile users, but it gives the attacker the possibility to perform statistical analysis and infer, eventually, the identities of the users. Finally, in the third case (case C) the attacker is not given any advantage on the privacy-preserving system, hence the attacker cannot link pseudonyms. In this case the attacker cannot know how many users are active and which applications are running on the same device. The attacker only learns which applications are used by the group of the users. Note that if the number of users in the group grows considerably, then the significance of this information diminishes.

It is worth noticing that the anonymity level depends on the number of users in the system, i.e. if only one user exists then the attacker can always infer who is behind a mobile app activity. On the other hand, if there is a large group of active users, then it is more difficult for the attacker to guess who is using a specific mobile app. This fact is highlighted in [39], which defines the concept of anonymity set as the set of participants from which a specific user could be confused with, and evaluates the anonymity level as the set size. Following this approach, in [40] authors provide a definition of anonymous state in terms of the state of not being identifiable within a set of subjects, i.e. the anonymity set. Authors also claim that the anonymity grows in relation to the set size, and it is higher when the probability of the set members to be active is evenly distributed. In [41] the authors continue towards this direction and propose a metric to quantify the anonymity set size that is based on the entropy of the system. These three works have been followed in subsequent works related to anonymity and privacy preservation, since they provide a useful definition of anonymity state and a metric to quantify the anonymity level.

V. System Requirements

The proposed privacy-enhanced OAuth 2.0 based protocol integrates a pseudonym-based signature scheme and

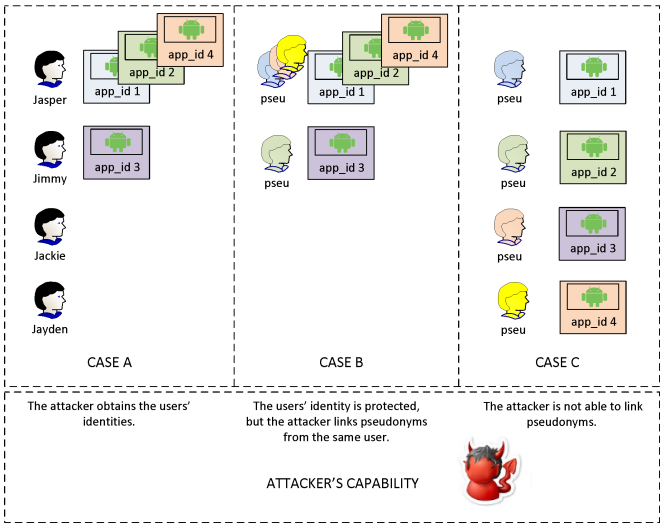

Fig. 4. Information obtained by the attacker in a toy scenario with 4 users and 4 running mobile apps. From left to right: the attacker can link pseudonyms to users; the attacker links pseudonyms from the same user; and the attacker is not given any advantage.

a signature delegation scheme into the user authentication process in the Step 2 of the OAuth 2.0 protocol flow with authorization code grant type. The proposed privacy-enhanced OAuth 2.0 based protocol also includes two new entities: i) the P-App installed in the user's mobile device; ii) and the Privacy Server which can be accessed remotely by the users and the AS. The proposed protocol's requirements are described below and derived from the different vulnerabilities described in the threat model:

1) *User privacy.* The identification of the user should only be possible by the AS when authorized by the user. Any other entity provided with the transcript of messages of one or several OAuth 2.0 protocol executions from the same user should infer no information about the user identity.
2) *Controlled user identification.* After a successful OAuth 2.0 protocol execution with a user, the AS should be able to obtain, through the PS, the relationship between the mobile app ID, the pseudonym and the real identity of the user.
3) *Mutual authentication between user and AS.* When the user authorizes an AS to obtain his/her real identity from the PS, the user should able to authenticate the AS, i.e. the attacker should not be able to impersonate the AS during the OAuth 2.0 protocol execution.
4) *Activity unlinkability.* The attacker should not be able to distinguish if several transcripts of the OAuth 2.0 protocol execution for different mobile applications belong to the same user or to different users, i.e. pseudonyms for different applications and users should be unlinkable.
5) *No sensitive information in the P-App.* Any malicious entity able to inspect the code of the P-App installed in the user's mobile device should get no advantage to perform any attack on the user's privacy.
6) *Low-complexity in the user side.* The protocol

should be performed in a useful time window in the user side. The complexity of the cryptographic operations in the user's mobile device should be low.

VI. Proposed privacy-enhanced OAuth 2.0 based protocol

The proposed protocol is based on the OAuth 2.0 protocol with authorization code grant type and improves the user privacy by integrating two cryptographic schemes into the user authentication process in Step 2 of the standard version of the OAuth 2.0 protocol with authorization code grant type. According to the proposed protocol, user authentication towards an AS is achieved without exposing the user credentials or any sensitive information towards an attacker controlling the mobile application or the browser, or eavesdropping the communication between the user and the AS. With the proposed protocol, users can self-generate pseudonyms for each different mobile application and get privately authenticated. Eavesdroppers are not able to link the users' pseudonyms to the users' identities, and are also not able to distinguish which pseudonyms belong to the same user. Moreover, although the proposed protocol requires to be assisted by the P-App in the user side, this P-App does not store any sensitive information and does not need to be tamper resistant.

We integrate two signature schemes into the OAuth 2.0 protocol flow, namely a pseudonym-based signature scheme and a delegation signature scheme. These schemes are adaptations from the ones proposed in [35] and our work in [42][36]. Although in this work we provide a detailed security analysis of the proposed protocol, we would like to refer interested readers to the works [35] and [42] for the formal security proofs of the proposed signature schemes.

A. System initialization

System initialization requires users and ASs to perform a registration in the Smart City PS, as depicted in Fig. 5. Users have to provide their real identities and select an nickname and password ($nick$, psw), which will be used to identify the user by the PS during the privacy-preserving user authentication, and to access their personal accounts in the PS. In these personal accounts, the user sets the personal information, e.g. user's real identity that is willing to share with other ASs. The PS will disclose this information to an AS if the AS is granted permission by the user during the execution of the proposed privacy-preserving OAuth 2.0 protocol.

The ASs also have to register in the PS and provide a list of users (their real identities) with personal accounts in such AS, i.e. the list of resource owners that are authenticated through that AS to access one or several resource servers. The PS maps the registered users in the PS to the list of users provided by the ASs. The AS can also provide a list of mobile apps that users may use to access such AS. This list can be used by the PS to pre-compute the pseudonym lists as described in section VIII.

Upon registration with the PS, the users and the ASs obtain the public values generated by the PS. The users and the ASs also obtain their specific values, i.e. the users obtain the static pseudonym and the ASs obtain their public pseudonym and corresponding credential. These values are detailed in the following sections for the different steps: i) parameter generation; ii) credential generation; and iii) static pseudonym generation. At the end of these three steps, the PS, all ASs registered to the PS, and the users registered to the PS obtain the secret and public values summarized in table I.

In the user side, the values obtained from the PS are stored in the P-App (except the user's nickname and password), which are used to perform the crytographic operations during the proposed privacy-enhanced OAuth 2.0 based protocol. Concretely, the P-App stores the PS public parameters, the ASs public pseudonyms, and the user's static pseudonym as a secret value. It is worth commenting that the disclosure of such static pseudonym would not give any advantage to an attacker to jeopardize user's privacy, it is only considered a secret value because it is never transmitted or disclosed during the proposed OAuth 2.0 protocol execution.

TABLE I. Protocol initialization values

Entity	Public Values	Secret Values
PS	Public Parameters G_1, G_2, P, H_1, H_2, H_3, $e()$, W, W_s	Secret key s
AS	Static pseudonym $pseu_v$	Credential Cre_v
P-App (user)	None	Static pseudonym $pseu_u$
User	None	Nick & Pswd $(nick_u, psw_u)$

1) Parameters generation: The PS performs the following operations to obtain the set of public parameters which are then distributed by the PS to the registered ASs and users:

- Selects two cyclic groups of prime order p, G_1 and G_2, where p is a prime of K bits. G_1 contains points in a elliptic curve where the ECDLP is hard [29]. Concrete implementation details are given in section VIII.
- Picks a point $P \in G_1$ as a generator.
- Selects a bilinear map e such that $e : G_1 \times G_1 \rightarrow G_2$.
- Selects two cryptographic hash functions $H_1, H_2 : \{0,1\}^* \rightarrow G_1$. [2]
- Selects another hash function $H_3 : \{0,1\}^* \rightarrow Z_p$.
- Picks randomly a secret value $s \in_R Z_p$ and generates a public key $W = sP$.
- Chooses randomly a public value $Q_s \in_R G_1$ and obtains a restriction key $W_s = sQ_s$.

2) Credential generation: An AS obtains a credential after registration with the PS. For credential generation for an AS v the PS performs the following steps:

[2] In a practical scenario it is selected a hash function $H : \{0,1\}^* \rightarrow A$, for any set A, and an admissible encoding function $L : A \rightarrow G_1$ [29].

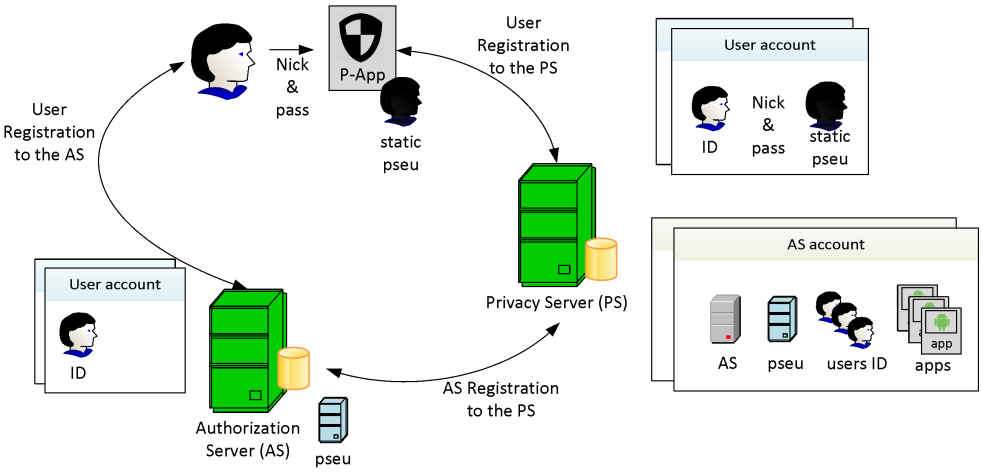

Fig. 5. Users and ASs registration process.

- Picks a value $\mu_v \in_R Z_p$.
- Computes a secret key $S_v = P\frac{1}{(s+\mu_v)}$.
- Sends to the AS v, over a secure channel, the credential $Cre_v = (\mu_v, S_v)$. The AS v verifies that $e(\mu_v P + W, S_v) = e(P, P)$ holds and accepts the credential.

3) Static pseudonyms generation: An AS v having a valid credential $Cre_v = (\mu_v, S_v)$ issued by the PS can generate a static pseudonym by performing the operation $pseu_v = \mu_v Q_s$. This pseudonym is public and unique of the AS v. Similarly, a citizen/user u also obtains a static pseudonym $pseu_u = \mu_u Q_s$, where μ_u is a secret value obtained from the user values $nick_u$ and psw_u: $\mu_u = H_3(nick_u||psw_u)$[3]. The value $pseu_u$ is stored in the P-App.

B. Privacy-enhanced user authentication

In the proposed privacy-enhanced OAuth 2.0 based protocol, the Step 2 maintains the same protocol flow as described in the original OAuth 2.0 protocol with authorization code grant type specification, but includes in the exchanged data the pseudonym-based signatures. The proposed Step 2 modification is described below in the steps: *UserAuthRequest1*; *UserAuthRequest2*; *UserAuthReponse1*; *UserAuthReponse2*; and *UserAuthentication*. The information used/generated during the privacy-enhanced user authentication is summarized in table II.

UserAuthRequest1 This step is performed by the AS. The AS v to the user u an authentication request code. The transmitted message is composed of the AS public pseudonym $pseu_v$, a pseudonym-based signature $sig_{pseu_v}(M)$ on a message M, and a random value rn. In this step the AS v:

- Obtains $Q_i = H_1(app\_id)$ from the mobile application specific identifier app_id.
- Obtains an app-specific pseudonym $pseu_{v,i} = \mu_v Q_i$ and a random value rn.
- Signs the message M, where $M = rn||pseu_{v,i}$, with its public pseudonym $pseu_v$. This signature operation is performed with signing algorithm described below.
- Sends M and the signature $sig_{pseu_v}(M)$ to the user.

To perform the signing algorithm referred above, for a message M with a pseudonym $pseu_v$, the AS v:

- Picks randomly α, r, $r' \in_R Z_p$.
- Obtains $T = \alpha S_v$, $R_{G_1} = rQ_i$ and $R = e(Q_i, P)^{r'}$.
- Obtains $c = H_3(M||T||R_{G_1}||R||pseu_v)$.
- Obtains $z_1 = c\alpha + r'$ and $z_2 = c\mu_v + r$.
- The signature $sig_{pseu_v}(M)$ is composed of the tuple (T, c, z_1, z_2).

UserAuthRequest2 The browser in the user side receives the authentication request code which is passed to the P-App for validation. To validate the code the P-App performs the signature verification algorithm on the signature $sig_{pseu_v} = (M)$, of the Message M, which is composed of the following steps:

- Obtain $R'_{G_1} = z_2 Q_s - pseu_v c$ and $R' = e(Q_s, P)^{z_1}/e(pseu_v + W_s, T)^c$. These operatons require the public values Q_s and W_s stored in the P-App.
- Obtain $c' = H_3(M||T||R'_{G_1}||R'||pseu_v)$.
- The verification is successful if the equality $c' = c$ holds.

After signature verification is successful the requuest code is validatied and the P-App prompts a user interface where the user u can type his/her nickname and password to construct the UserAuthResponse1.

UserAuthReponse1 The P-App asks the user u to insert the nickname and password $(nick_u, psw_u)$ and obtains user's static pseudonym value $pseu_u = \mu_u Q_s$, where $\mu_u = H_3(nick_u||psw_u)$. The P-App verifies that the calculated pseudonym matches the stored pseudonym obtained during the user registration with the PS. If the pseudonym matches then the P-App:

[3]The operator || refers to a string concatenation

TABLE II. PROTOCOL EXECUTION VALUES

Entity	Public Values	Secret Values
PS	Public Parameters G_1, G_2, P, H_1, H_2, H_3, $e()$, W, W_s	Secret key s
AS	Pseudonyms $pseu_v$, $pseu_{v,i}$; Signatures $sig_{pseu_u}(M)$, $Delsig'_{pseu_{v,i},pseu_{u,i}}(R, w_u)$, nonce rn	Credential Cre_v
P-App (user)	App_ID app_id; App-specific pseudonym $pseu_{u,i}$; Warrant w_u; Signature $sig_{pseu_u}(w_u)$	Static pseudonym $pseu_u$
User	None	Nick & Pswd $(nick_u, psw_u)$

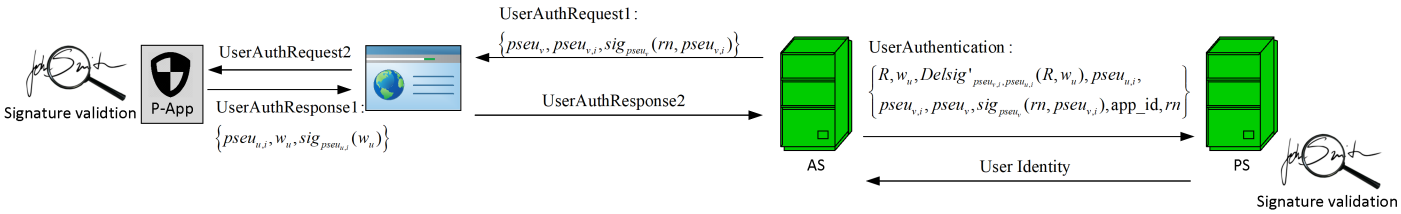

Fig. 6. Privacy-enhanced user authentication of the OAuth 2.0 proposed protocol.

- Uses the application ID to obtain the value $Q_i = H_1(app\_id)$.
- Obtains the app-specific pseudonym $pseu_{u,i} = \mu_{u,i}Q_i$, where $\mu_{u,i} = H_3(nick_u||psw_u||app\_id)$. The pseudonym is app-specific because the user u obtains different pseudonyms for other application IDs. If the user tries to use the same pseudonym for other applications the authentication will fail in the PS side. It is also worth mentioning that the app-specific pseudonym of a user is different than that of other users for the same application, thus the pseudonyms are app-specific and user-specific.
- Signs a warrant w_u, where w_u is the tuple $(pseu_{v,i}||rn||D)$ and D is the delegation scope (time of validity), with the pseudonym $pseu_{u,i}$. The signature, $sig'_{pseu_{u,i}}(w_u)$, is performed with a delegation signing algorithm consisting of the following operation:
  - $sig'_{pseu_{u,i}}(w_u) = \mu_{u,i}H_1(w_u)$.
- Constructs the tuple $Del_{u,v} = (pseu_{u,i}, w_u, sig'_{pseu_{u,i}}(w_u))$.
- The value $Del_{u,v}$ is sent back to the AS v through the browser as the authentication response code. It is worth mentioning that this code cannot be replayed in a later occasion. Also, it does not provide any useful information to identify the user or to link this user to other OAuth 2.0 authentications of the same user for other mobile applications.

UserAuthResponse2 The AS v receives the authentication response code, consisting of the tuple $Del_{u,v} = (pseu_{u,i}, w_u, sig'_{pseu_{u,i}}(w_u))$, and obtains a delegated signature. To obtain a delegated signature the AS v:

- Generates a request R, where the AS asks the PS for the real identity of the user holding the pseudonym $pseu_{u,i}$.
- Computes:

$$Delsig'_{pseu_v,pseu_u}(R, w_u) = sig'_{pseu_u}(w_u) + \mu_v H_2(R||w_u). \quad (1)$$

UserAuthentication The AS contacts the PS authenticate the user and identify the user account in the AS, i.e. to identify the user. With this objective, the AS sends the combined signature with the request for the user identity to the PS, which consists of the following tuple: R, w_u, $Delsig'_{pseu_{v,i},pseu_{u,i}}(R, w_u)$, $pseu_{u,i}$, $pseu_{v,i}$, $pseu_v$, app_id. All these values where obtained in the previous steps.

Upon reception of the message from the AS, the PS:

- Verifies that the pseudonym $pseu_v$ is a public pseudonym of a valid AS server.
- Obtains the app-specific value $Q_i = H_1(app\_id)$ using the application ID app_id.
- Verifies that the app-specific pseudonym presented by the AS, i.e. $pseu_{v,i}$, is linked to the static pseudonym $pseu_v$ by verifying that $e(Q_s, pseu_{v,i}) = e(pseu_v, Q_i)$. (It is worth mentioning that this step cannot be done for the user's pseudonyms since they are unlinkable).
- Validates the delegated signature, $Delsig'_{pseu_{v,i},pseu_{u,i}}(R, w_u)$, by performing checking whether the following equation holds:

$$e(Delsig'_{pseu_{v,i},pseu_{u,i}}(R, w_u), Q_i) = e(H_1(w_u), pseu_{u,i})e(H_2(R||w_u), pseu_{v,i}) \quad (2)$$

- Identifies the user u by linking the pseudonym $pseu_{u,i}$ to the user real identity. For that, the PS obtains the app-specific pseudonyms for all the users included in the list of users of the AS v, i.e. computes $pseu_{x,i} = \mu_{x,i}Q_i$, for every user x and for $Q_i = H_1(app\_id)$, and selects the user for which $pseu_{x,i} = pseu_{u,i}$. Note that, as detailed in section VIII, the PS can pre-compute and store the app-specific pseudonyms for the users of each AS and for different app_ids, to speed up the look up process.

After identifying the user, the PS sends a message over a secure channel to the AS v with the user's identity. The

proposed privacy-enhanced user authentication is summarized in Fig. 6.

VII. Security analysis

This section provides the security analysis of the proposed protocol and the correctness of the cryptographic operations performed. The security analysis also counts with an implementation in Proverif to evaluate the reachability properties of the secret values used to obtain pseudonyms in the user side, and the observational equivalence of the signatures obtained with pseudonyms from different users. The proverif code for the evaluation of observational equivalence can be found in the annex. We woul like to refer insterested readers to the works [35] and [42] for the formal security proofs.

A. Correctness of the signature and delegation schemes

The correctness of the pseudonym-based signature scheme used the UserAuthRequest can be found in [35], hence it is not included in this section. In this section we provide the correctness of the delegated signature used in the UserAuthResponse. It is worth commenting that this type of delegated signature scheme was first presented in [43].

The signature, $Delsig'_{pseu_v,pseu_u}(R, w_u)$, is validated with the equation:

$$\begin{aligned} &e(Delsig'_{pseu_{v,i},pseu_{u,i}}(R, w_u), Q_i) \\ &= e(H_1(w_u), pseu_{u,i})e(H_2(R||w_u), pseu_{v,i}) \end{aligned} \tag{3}$$

with some mathematical manipulation, following the properties of bilinear maps detailed in section XI:

$$\begin{aligned} e(H_1(w_u), pseu_{u,i})e(H_2(R||w_u), pseu_{v,i}) = \\ e(H_1(w_u), Q_i)^{\mu_u} e(H_2(R||w_u), Q_i)^{\mu_v} = \\ e(H_1(w_u)\mu_u, Q_i)e(H_2(R||w_u)\mu_v, Q_i) = \\ e(H_1(w_u)\mu_u + H_2(R||w_u)\mu_v, Q_i) = \\ e(Delsig'_{pseu_{v,i},pseu_{u,i}}(R, w_u), Q_i) \end{aligned} \tag{4}$$

B. Conditional privacy

Conditional privacy is achieved if only the PS can link the pseudonyms to users identity. Any other entity cannot link the pseudonym of a user to its identity or to other pseudonyms of the same user. Note that a user's pseudonym is constructed as $pseu_{u,i} = \mu_{u,i}Q_i$ for the app-specific pseudonyms and $pseu_u = \mu_u Q_s$ for the static pseudonym. Although the values Q_s and Q_i are public, the μ_u and $mu_{u,i}$ values are secret and cannot be obtained from the pseudonyms since ECDLP is hard in G_1. The PS however can compute any app-specific pseudonym generated in the user side since it poses the required values $nick_u$, psw_u and app_id.

Finally, we have tested in Proverif the capability of the attacker to steal the user μ values and we have verified that the attacker has no access to such values in a scenario with several users interacting with an AS, the result of the Proverif [19] outcome is presented below:

```
-- Query not attacker(mu1[])
Initial clauses:
Clause 0: attacker(v_118) && attacker(v_119)
&& attacker(v_120) && attacker(v_121)
-> attacker(sign(v_118,v_119,v_120,v_121))
(The attacker applies function sign.)

Clause 1: attacker(v_124) && attacker(v_125)
-> attacker(e(v_124,v_125))
(The attacker applies function e.)

[...]

Clause 43: attacker(app_id1_631)
-> attacker(sign((app_id1_631,
mulZpG1(mv[],H1(app_id1_631)),rn1_680),
Qs[],sk(P_103[],mv[],s_102[]),
mulZpG1(mv[],Qs[])))
(If the message app_id1_631 is received
from the attacker at input {24},
then the message sign((app_id1_631,mulZpG1
(mv[],H1(app_id1_631)),rn1_680),Qs[],
sk(P_103[],mv[],s_102[]),mulZpG1(mv[],Qs[]))
may be sent to the attacker at output {31}.)
Abbreviations:
rn1_680 = rn1_105[app_id1_104 = app_id1_631,
!1 = @sid_632]

Completing...
Starting query not attacker(mu1[])
RESULT not attacker(mu1[]) is true.
```

As indicated in [31], an attacker can track user's activities and infer the user's identity if the same pseudonym is always used, hence in a mobile scenario users should be able to generate a large number of pseudonyms and be provided with an efficient mechanism to switch the active pseudonym. We address this issue by providing different pseudonyms per application, which are generated on demand in the user side. These pseudonyms are unlinkable hence preventing eavesdroppers from linking different activities, i.e. mobile applications, to the same user. It is worth mentioning that the system requirements (sec. V) of user privacy and user identification hold, since only the AS can obtain the user's identity though the PS when authorized by the user. Also, the activity unlinkability requirement is fulfilled since the user shows different unlinkable pseudonyms when using different mobile apps.

We have tested the privacy properties in Proverif and we have obtained verification that the attacker is not capable of distinguishing a scenario of one user authenticating with two distinct mobile application and two users using one mobile application each. The result of the Proverif outcome is presented below and the source code of the Proverif implementation is given in the Appendix.

```
-- Observational equivalence
Initial clauses:
Clause 0: v_133 <> true &&
attacker2(v_133,true) &&
attacker2(@mayfail_134,@mayfail_135)
-> attacker2(false,@mayfail_135)
(The attacker applies function &&.)
```

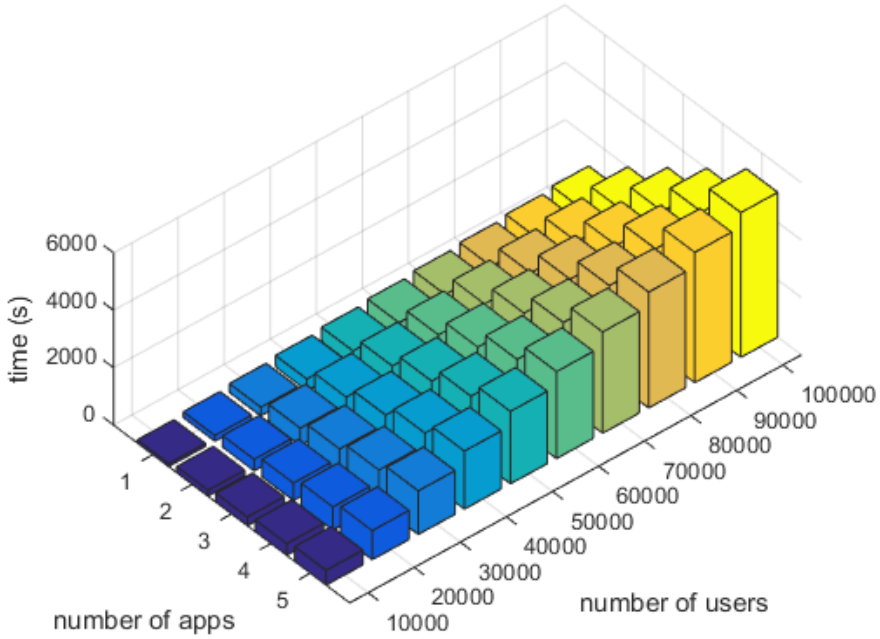

Fig. 7. Time required for pseudonym pre-computation of up to 100K users and 5 apps.

```
Clause 1: v_139 <> true &&
attacker2(true,v_139) &&
attacker2(@mayfail_138,@mayfail_140)
-> attacker2(@mayfail_138,false)
(The attacker applies function &&.)

[...]

Clause 106: attacker2((P_103[],Qs[],
mulZpG1(s_102[],Qs[]),mulZpG1(s_102[],
P_103[])),(P_103[],Qs[],mulZpG1(s_102[],Qs[]),
mulZpG1(s_102[],P_103[])))
(The message (P_103[],Qs[],
mulZpG1(s_102[],Qs[]),mulZpG1(s_102[],P_103[]))
(resp. (P_103[],Qs[],mulZpG1(s_102[],Qs[]),
mulZpG1(s_102[],P_103[])))
may be sent to the attacker at output {12}.)

200 rules inserted. The rule base
contains 180 rules. 67 rules in the queue.

[...]

6600 rules inserted. The rule base
contains 6168 rules. 50 rules in the queue.

RESULT Observational equivalence is true
(bad not derivable).
```

C. Autonomy

It is worth noting that when users download new applications the users are not required to contact the PS to obtain new app-specific pseudonyms since the pseudonyms are self-generated. The user identification in the PS side is still achievable through the proposed anonymity revocation mechanism. Thus, in the proposed protocol the users are only required to contact the PS to register.

D. Mutual Authentication

As specified in sec. V, the proposed protocol provides mutual authentication between the user and the AS, since the P-App authenticates the AS. The AS is registered in the PS and has a public pseudonym $pseu_v$, which is also stored in the P-App in the user side.

E. P-App Tampering resiliency

The P-App does not hold any sensitive information about the user, and the leaking of any value stored by the PS can never lead to user tracking or user impersonation. The static pseudonym stored in the P-App to check the validity of the user credentials is not linkable to the different app-specific pseudonyms used by the user during the OAuth 2.0 protocol execution. Hence, following the system requirements, sec. V, the P-App does not need to be tamper resistant.

VIII. Performance Evaluation

To evaluate the performance of the proposed protocol, we implemented the signature schemes and the revocation scheme in JAVA using the library provided in [44]. The tests performed evaluate the computation time and memory of the more demanding parts of the proposed protocol in the user and server sides. We have chosen a Type A curve with the recommended settings in [45], i.e. $q = 512$ bits and $r = 160$ bits (G_1 is a cyclic group, with prime order of 160bits, of points in a elliptic curve with points of 1024 bits). We have tested the performance in terms of

execution time of the different steps of the protocol, with special focus on the anonymity revocation step in the PS side and the signature verification in the user side.

It is worth noting that the pseudonyms are self-generated in the user side, thus the PS has to compute the pseudonyms for all users and all mobile apps per AS in order to match an unknown pseudonym with its owner and retrieve the user identity. This operation can take considerable time if the number of users registered in the AS is considerable, and it also increases with the number of mobile apps that access the AS. Hence, the pseudonym list should be pre-computed at the PS before protocol execution. We have performed tests to evaluate the computation time of the pseudonym list at the PS side. Figure 7 shows the time lapsed in the pre-computation of the list of pseudonyms for the PS with up to 100K registered users and up to 5 mobile apps.

It can be appreciated that the time complexity grows linearly with the number of users and apps. For the case of 5 apps and 100K users, the computation time takes 1 hour and 24 minutes. However, saving and loading this 500K pseudonyms in a file takes less than 2 seconds (1.56 and 1.82 respectively, 5 more seconds are needed if the psedonyms are stored in bitstrings due to type conversion) and requires 64MB of memory. For the concrete case of 1 mobile app and 100K users, the computation time is less than 17 minutes, whereas the reading time from a file is in the order of 1300ms including the type conversion. Although not shown in the figure, we have also tested the creation of up to 3 million pseudonyms, yielding a computation time of 8h 38min and 384MB space.

In the user side, that has been implemented in an Android application (P-App), the signature verification takes 90ms, since it involves a bilinear pairing calculation. Both the pseudonym generation and the delegatable signature take 50ms. However the computation time can very significantly depending one the mobile device. When tested in an android device with low processing capabilities, signature verification has gone up to 1200ms and pseudonym generation and the delegatable signature has gone up to 500ms. In the server side, the signature takes 60ms, since it is more complex than the signature performed in the user side (this signature performed in a hardware limited android device would require 700ms, however it is not required in our proposed protocol).

IX. Conclusion

This paper addresses the previously reported privacy issues of OAuth 2.0 protocol with authorization code grant type by integrating a pseudonym-based signature scheme and a delegation signature scheme into the OAuth 2.0 protocol flow. Concretely, the proposed modification only affects the exchanged data of the Step 2 in the OAuth 2.0 protocol with authorization code grant type. The proposed solution allows users to authenticate and authorize different mobile application under different unlinkable pseudonyms, which prevents eavesdroppers from tracking users' activities. Moreover, the application specific pseudonyms are generated on demand without the participation of the trusted authority (i.e. the Privacy Server in the proposed system model). The proposed solution does not require the storage of sensitive information in the mobile device, hence the proposed Privacy Application in the user side does not need to be tamper resistant.

References

[1] United Nations. Department of Economic and Social Affairs. *World Urbanization Prospects, the 2014 Revision: Highlights*. 2014.

[2] ITU. An overview of smart sustainable cities and the role of information and communication technologies. 2014.

[3] Milind Naphade, Guruduth Banavar, Colin Harrison, Jurij Paraszczak, and Robert Morris. Smarter cities and their innovation challenges. *Computer*, 44(6):32–39, June 2011.

[4] Hans Schaffers, Nicos Komninos, Marc Pallot, Brigitte Trousse, Michael Nilsson, and Alvaro Oliveira. The future internet. chapter Smart Cities and the Future Internet: Towards Cooperation Frameworks for Open Innovation, pages 431–446. Springer-Verlag, Berlin, Heidelberg, 2011.

[5] Hafedh Chourabi, Taewoo Nam, Shawn Walker, J. Ramon Gil-Garcia, Sehl Mellouli, Karine Nahon, Theresa A. Pardo, and Hans Jochen Scholl. Understanding smart cities: An integrative framework. In *Proceedings of the 2012 45th Hawaii International Conference on System Sciences*, HICSS '12, pages 2289–2297, Washington, DC, USA, 2012. IEEE Computer Society.

[6] M. Fengou, G. Mantas, D. Lymberopoulos, N. Komninos, S. Fengos, and N. Lazarou. A new framework architecture for next generation e-health services. *Biomedical and Health Informatics, IEEE Journal of*, 17(1):9–18, Jan 2013.

[7] D. Hardt. The oauth 2.0 authorization framework. In *IETF2012*, 2012.

[8] G. Sciarretta, R. Carbone, and S. Ranise. A delegated authorization solution for smart-city mobile applications. In *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 1–6, Sept 2016.

[9] Eric Y. Chen, Yutong Pei, Shuo Chen, Yuan Tian, Robert Kotcher, and Patrick Tague. Oauth demystified for mobile application developers. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 892–903, New York, NY, USA, 2014. ACM.

[10] M. Shehab and F. Mohsen. Towards enhancing the security of oauth implementations in smart phones. In *MS, 2014 IEEE*, June.

[11] M. M. T. Lodderstedt and P. Hunt. Oauth 2.0 threat model and security considerations. In *IETF2013*, 2013.

[12] Mariantonietta La Polla, Fabio Martinelli, and Daniele Sgandurra. A survey on security for mobile devices. *IEEE Communications Surveys and Tutorials*, 15(1):446–471, 2013.

[13] Mantas G, Komninos N, Rodriguez J, Logota E, and Marques H. Security for 5g communications. In *Fundamentals of 5G Mobile Networks*. John Wiley & Sons, 2015.

[14] A. Arabo and B. Pranggono. Mobile malware and smart device security: Trends, challenges and solutions. In *2013 19th International Conference on Control Systems and Computer Science*, pages 526–531, May 2013.

[15] M. Becher, F. C. Freiling, J. Hoffmann, T. Holz, S. Uellenbeck, and C. Wolf. Mobile security catching up? revealing the nuts and bolts of the security of mobile devices. In *2011 IEEE Symposium on Security and Privacy*, pages 96–111, May 2011.

[16] A. Martinez-Balleste, P. A. Perez-martinez, and A. Solanas. The pursuit of citizens' privacy: a privacy-aware smart city is possible. *IEEE Communications Magazine*, 51(6):136–141, June 2013.

[17] B. Krupp, N. Sridhar, and W. Zhao. Spe: Security and privacy enhancement framework for mobile devices. *IEEE Transactions on Dependable and Secure Computing*, PP(99):1–1, 2015.

[18] V. Sucasas, G. Mantas, A. Radwan, and J. Rodriguez. An oauth2-based protocol with strong user privacy preservation for smart city mobile e-health apps. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6, May 2016.

[19] Bruno Blanchet and Ben Smyth. Proverif: Automatic cryptographic protocol verifier user manual & tutorial (2011).

[20] E. Hammer-Lahav. The oauth 1.0 protocol. In *IETF*, 2010.

[21] San-Tsai Sun and Konstantin Beznosov. The devil is in the (implementation) details: An empirical analysis of oauth sso systems. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 378–390, New York, NY, USA, 2012. ACM.

[22] Daniel Fett, Ralf Küsters, and Guido Schmitz. A comprehensive formal security analysis of oauth 2.0. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1204–1215, New York, NY, USA, 2016. ACM.

[23] C. Bansal, K. Bhargavan, and S. Maffeis. Discovering concrete attacks on website authorization by formal analysis. In *2012 IEEE 25th Computer Security Foundations Symposium*, pages 247–262, June 2012.

[24] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, October 1985.

[25] Dijiang Huang. Pseudonym-based cryptography for anonymous communications in mobile ad hoc networks. *Int. J. Secur. Netw.*, 2(3/4):272–283, April 2007.

[26] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT '01, pages 93–118, London, UK, UK, 2001. Springer-Verlag.

[27] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *Proceedings of the 6th Annual International Workshop on Selected Areas in Cryptography*, SAC '99, pages 184–199, London, UK, UK, 2000. Springer-Verlag.

[28] Maxim Raya and Jean-Pierre Hubaux. The security of vehicular ad hoc networks. In *Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks*, SASN '05, pages 11–21, New York, NY, USA, 2005. ACM.

[29] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. *SIAM J. of Computing*, 32(3):586–615, 2003. extended abstract in Crypto'01.

[30] Yanchao Zhang, Wei Liu, and Wenjing Lou. Anonymous communications in mobile ad hoc networks. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 3, pages 1940–1951 vol. 3, March 2005.

[31] Rongxing et al. A lightweight conditional privacy-preservation protocol for vehicular traffic-monitoring systems. *Intelligent Systems, IEEE*, 28(3):62–65, May 2013.

[32] Yipin Sun et al. An efficient pseudonymous authentication scheme with strong privacy preservation for vehicular communications. *Vehicular Technology, IEEE Transactions on*, 59(7):3589–3603, Sept 2010.

[33] L. L. Wang, G. Z. Liu, L. j. Sun, and Y. W. Lin. An effective pseudonym generating scheme for privacy and anonymity in vanets. In *2016 International Conference on Information System and Artificial Intelligence (ISAI)*, pages 267–270, June 2016.

[34] Giorgio Calandriello, Panos Papadimitratos, Jean-Pierre Hubaux, and Antonio Lioy. Efficient and robust pseudonymous authentication in vanet. In *Proceedings of the Fourth ACM International Workshop on Vehicular Ad Hoc Networks*, VANET '07, pages 19–28, New York, NY, USA, 2007. ACM.

[35] Yong Zhang and Jun-Liang Chen. A delegation solution for universal identity management in soa. *Services Computing, IEEE Transactions on*, 4(1):70–81, Jan 2011.

[36] Victor Sucasas, Georgios Mantas, Firooz B. Saghezchi, Ayman Radwan, and Jonathan Rodriguez. An autonomous privacy-preserving authentication scheme for intelligent transportation systems. *Comput. Secur.*, 60(C):193–205, July 2016.

[37] P. Hunt T. Lodderstedt, M. McGloin. Oauth 2.0 threat model and security considerations. RFC 6819, Internet Engineering Task Force (IETF), January 2013.

[38] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[39] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptol.*, 1(1):65–75, March 1988.

[40] Andreas Pfitzmann and Marit Hansen. Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management - a consolidated proposal for terminology, February 2008.

[41] Andrei Serjantov and George Danezis. *Towards an Information Theoretic Metric for Anonymity*, pages 41–53. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[42] V. Sucasas, F. Saghezchi, A. Radwan, H. Marques, J. Rodriguez, S. Vahid, and R. Tafazolli. Efficient privacy preserving security protocol for vanets with sparse infrastructure deployment. In *ICC, IEEE*, 2015.

[43] Huang et al. A short proxy signature scheme: Efficient authentication in the ubiquitous world. In Tomoya Enokido, editor, *EUC Workshops*, volume 3823, pages 480–489. Springer, 2005.

[44] Angelo De Caro and Vincenzo Iovino. jpbc: Java pairing based cryptography. In *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*, pages 850–855, Kerkyra, Corfu, Greece, June 28 - July 1, 2011. IEEE.

[45] Ben Lynn. *On the Implementation of pairing based cryptosystems.* PhD Thesis, Stanford University, 2007.

X. Acknowledgments

This work is supported by the European Regional Development Fund (FEDER), through the Regional Operational Programme of Centre (CENTRO 2020) of the Portugal 2020 framework [Project PRIVACITIES with Nr. 017785 (CENTRO-01-0247-FEDER-017785)], and has also been carried out in the scope of the project labelled as CAT209-H2O (Human to Objects - Easy Interactions in the Smart City) by the European EUREKA-CATRENE programme.

XI. ANNEX: Preliminaries

This section provides the mathematical preliminaries required for the Understanding of the proposed privacy-enhanced OAuth 2.0 based protocol, and the definition of the mathematical problems in which the security of the proposed solution relays on, i.e. the Collusion Attack Algorithm with k Traitors and n Examples ((k,n)-CAA) [35], and the Elliptic Curve Discrete Logarithm Problem (ECDLP).

A. Bilinear Maps

Let G_1 and G_2 be two groups of prime order r and security parameter K (defining the bits of r). The discrete logarithm problem is hard in both groups. Then e is a bilinear map in the groups (G_1, G_2), $e : G_1 \times G_1 \rightarrow G_2$ if it satisfies the following properties [29]:

- bilinearity: $\forall \alpha, \beta \in Z_p^*$ and $P, Q, R \in G_1$, we have that $e(\alpha P + \beta Q, R) = e(P, R)^\alpha e(Q, R)^\beta$ and $e(R, \alpha P + \beta Q) = e(R, P)^\alpha e(R, Q)^\beta$.
- no-degeneracy: there is at least one element $Q \in G_1$ such that $e(Q, Q) \neq 1_{G_2}$.
- complexity: The bilinear map e can be computed efficiently.

B. ECDLP

Given a group $G = \langle P \rangle$ of prime order p. Given any point $Q \in G$, determining an integer $n \in [1, p-1]$ such that $Q = nP$ is called the Elliptic Curve Discrete Logarithm Problem (ECDLP) and considered to be computationally intractable in polynomial time.

C. (k,n)-CAA Problem

Let G_1, G_2 and e be as above, and let k and n be integers; let $P, P_1, \ldots, P_n \in G_1$, and let $x, a_1, \ldots, a_k \in Z_p$. Then, as defined in [35] the Collusion Attack Algorithm problem with k traitors and n examples $((k, n) - CAA)$ is as follows:

Given the set $\{xP, xP_j | 1 \leq j \leq n\}$, there is not any polynomial time algorithm for obtaining $\frac{1}{(x+a)}P$ for some $aP_j \notin \{a_i P_j | 1 \leq i \leq k; 1 \leq j \leq n\}$ with no negligible probability.

D. Fiat-Shamir Heuristic

The Fiat-Shamir heuristic is a technique to convert an interactive proof of knowledge into a non interactive proof that can be used as a digital signature. First lets define an interactive proof of knowledge where a prover, Alice (A), intends to proof the knowledge of a secret value x to a verifier, Bob (B):

1) A has a secret x, such that $y = g^x$. g is a generator of a cyclic group G of prime order q where the discrete logarithm problem is hard, and y is the A's public value belonging to such group.
2) A selects $v \in_R Z_q$ and sends $t = g^v$ to B.
3) B selects a nonce $c \in_R Z_q$ and sends c to A.
4) A obtains $r = v - cx$ and sends r to B.
5) B verifies that $t = g^r y^c$ holds.

The Fiat-Shamir heuristic substitutes the nonce generation in the verifier side by a random generation in the prover side, hence avoiding the 3-way handshake and limiting the number of transmissions to only one. This only one message can be considered a digital signature. The random generation is performed with a random oracle, i.e. a hash function in a practical scenario. The non-interactive proof of knowledge is now described as follows:

1) A knows x such that $y = g^x$.
2) A picks $v \in_R Z_q$ and gets $t = g^v$, but A does not send t to B.
3) A generates $c = H(g, y, t)$ with a hash function $H()$ also known by B.
4) A obtains $r = v - cx$.
5) A sends (t, r) to B.
6) B can obtains $c = H(g, y, t)$ and verifies that $t = g^r y^c$ holds.

XII. APPENDIX: Proverif code

The code defines two processes, one representing two USERS and the other representing the AS. The process USERS receives two credentials (nickname and password), representing two different users. USERS performs the protocol together with the AS in a first iteration using the credential of the first user. In a second iteration the process USERS makes a random choice over which credential is used, and performs the authentication protocol for a different mobile application. Thus, the attacker is asked to distinguish the cases.

```
type G1.
type G2.
type Zp.

fun mulZpZp(Zp, Zp): Zp.
fun mulZpG1(Zp, G1): G1.
fun addZpZp(Zp, Zp): Zp.
fun addG1G1(G1, G1): G1.
fun exp(G2, Zp): G2.
fun mul(G2, G2): G2.
fun e(G1, G1):G2.
fun H1(bitstring):G1.
fun H2(bitstring):G1.
fun H3(bitstring):Zp.
fun sk(G1, Zp, Zp): G1.
fun sign(bitstring, G1, G1, G1):
bitstring.

reduc forall z: Zp, ga: G1, gb:G1;
pairing1(e(mulZpG1(z, ga), gb))
=exp(e(ga, gb),z).
reduc forall z: Zp, ga: G1, gb:G1;
pairing2(e(ga, mulZpG1(z, gb)))
=exp(e(ga, gb),z).

reduc forall za: Zp, zb: Zp, ga: G1,
gb:G1; pairing3(e(mulZpG1(za,ga),
mulZpG1(zb, gb)))
=exp(e(ga,gb), mulZpZp(za,zb)).

reduc forall M: bitstring, Q: G1,
P: G1, mu: Zp, s: Zp,  pseu: G1;
checksign(sign(M,Q, sk(P, mu, s),
mulZpG1(mu,Q)),M, Q, mulZpG1(mu,Q))
= true.

reduc forall M: bitstring, Q: G1,
S: G1, pseu: G1; getpseu(sign(M,Q, S, pseu))
= pseu.

reduc forall M: bitstring, Q: G1,
S: G1, pseu: G1; getmess(sign(M,Q, S, pseu))
= M.

free c1 : channel.
free c2 : channel.

let USERS(user: bitstring,
nick_pasw1: bitstring, nick_pasw2: bitstring,
P:G1, Qs:G1, Ws:G1, W:G1, pseu_v:G1) =
```

```
new app_id1: bitstring;
out(c2, app_id1);
in(c2,sig_v1:bitstring);
if pseu_v = getpseu(sig_v1) then
let M1 = getmess(sig_v1) in
let (=app_id1, pseu_vi1: G1, rn1: bitstring)
= M1 in
if checksign(sig_v1, M1, Qs, pseu_v)
= true then
let mu1 = H3((nick_pasw1,app_id1)) in
let Qi1 = H1(app_id1) in
let pseu_ui1 = mulZpG1(mu1, Qi1) in
let wu1 = (pseu_vi1, rn1) in
let sig_ui1 = mulZpG1(mu1, H1(wu1))
in out(c2, (sig_ui1, wu1, pseu_ui1));


new app_id2: bitstring;
out(c2, app_id2);
in(c2,sig_v2:bitstring);
let pseu_v2 = getpseu(sig_v2) in
let M2 = getmess(sig_v2) in
let (=app_id2, pseu_vi2: G1, rn2: bitstring)
= M2 in
if checksign(sig_v2, M2, Qs, pseu_v2)
= true then
let mu2 = H3((user,app_id2)) in
let Qi2 = H1(app_id2) in
let pseu_ui2 = mulZpG1(mu2, Qi2) in
let wu2 = (pseu_vi2, rn2) in
let sig_ui2 = mulZpG1(mu2, H1(wu2))
in out(c2, (sig_ui2, wu2, pseu_ui2)).

let AS(mv: Zp, Sv: G1, P:G1, Qs:G1,
Ws:G1, W:G1) =

in(c2,app_id1: bitstring);
new rn1: bitstring;
let pseu_v1 = mulZpG1(mv, Qs) in
let Qi1=H1(app_id1)   in
let pseu_vi1 = mulZpG1(mv, Qi1) in
let M1=(app_id1, pseu_vi1, rn1) in
let sign_v1 = sign(M1, Qs, Sv, pseu_v1)
in out(c2, sign_v1);
in(c2, sig_ui1:bitstring);
in(c2,app_id2: bitstring);
new rn2: bitstring;
let pseu_v2 = mulZpG1(mv, Qs) in
let Qi2=H1(app_id2)   in
let pseu_vi2 = mulZpG1(mv, Qi2) in
let M2=(app_id2, pseu_vi2, rn2) in
let sign_v2 = sign(M2, Qs, Sv, pseu_v2)
in out(c2, sign_v2);
in(c2, sig_ui2:bitstring).

process

new s:Zp;
new P:G1;
new Qs:G1;
new nick_pasw1: bitstring;
new nick_pasw2: bitstring;
new mv: Zp;
let Ws = mulZpG1(s, Qs) in
let W  = mulZpG1(s, P) in
let PPs = (P, Qs, Ws, W) in out(c1, PPs);
((let user = choice[nick_pasw1,nick_pasw2]
in USERS(user, nick_pasw1,
nick_pasw2,      P, Qs, mulZpG1(s, Qs),
mulZpG1(s, P),   mulZpG1(mv, Qs))) |
(let Sv = sk(P, mv,  s) in
AS(mv, Sv, P, Qs, mulZpG1(s, Qs),
mulZpG1(s, P))))
```

XIII. Biographies

Victor Sucasas obtained his Ph.D. on Electronic Engineering at University of Surrey (UK) in 2016. He has extensive research experience as a researcher at Instituto de Telecomunicações - Aveiro, Portugal and as a PhD student at University of Surrey, Guildford, UK, where he worked on European projects FP7-GREENET, ECSEL-SWARMs and CATRENE-H2O. Over the past 6 years, he has been an active researcher in several fields such as wireless cooperative communications, network security and privacy preserving systems.

Georgios Mantas received the Ph.D. degree in Electrical and Computer Engineering from the University of Patras, Greece, in 2012, the M.Sc. degree in Information Networking from Carnegie Mellon University, Pittsburgh, PA, in 2008, and the Diploma in Electrical and Computer Engineering from the University of Patras, Greece, in 2005. He is currently a Senior Researcher at the Instituto de Telecomunicações - Aveiro, Portugal, where he has been involved in research projects such as ECSEL-SemI40, CATRENE-MobiTrust, CATRENE-NewP@ss, ARTEMIS-ACCUS, FP7-CODELANCE, and FP7-SEC-SALUS. His research interests include network and system security, authentication mechanisms, privacy-preserving mechanisms, intrusion detection systems, and secure network coding.

Saud Althunibat is an Assistant Professor at the Department of Communications Engineering of Al-Hussein Bin Talal University, Jordan. He received the Ph.D. degree in Telecommunications in 2014 from the University of Trento, Italy. From 2011 to 2014, he has been a Marie-Curie researcher within the GREENET project at University of Trento. He is a reviewer in many international journals and a TPC member in many international conferences. He is the recipient of the best-paper award in IEEE CAMAD 2012, and was selected as exemplary reviewer in IEEE Communication Letters 2013. His research interests include Cognitive Radio Networks, Physical-Layer Security, Resource Allocation and Heterogeneous Networks.

Leonardo Oliveira is currently an undergraduate student at the Department of Electronics, Telecommunications and Informatics at the University of Aveiro. He is currently an active collaborator with the Instituto de Telecomunicações - Aveiro as a software developer for security systems.

Angelos Antonopoulos received his PhD degree from the Signal Theory and Communications (TSC) Department of the Technical University of Catalonia (UPC) in December 2012. His main research interests include cooperative communications, MAC protocols, network coding and energy efficient network planning. He has participated in several European and Spanish national projects (e.g., GREENET, GreenT, CO2GREEN) and has served as an expert evaluator of research projects for the National Council for Scientific Research. He has been granted three annual scholarships by the Greek State Scholarships Foundation (IKY) and, recently, he has been awarded the First Polytechnic Graduates Prize by the Technical Chamber of Greece (TEE-TGC).

Ifiok Otung is Professor of Satellite Communications at the University of South Wales (USW, formerly University of Glamorgan). He earned the degrees of BSc (First Class Honours) and MSc in Electronic & Electrical Engineering from the University of Ife Nigeria, and PhD in Satellite Communications from the University of Surrey, UK. Since 1997 he has been at USW where he teaches courses in Satellite, Mobile and Digital Communications and has supervised around 120 postgraduate projects, including MSc and PhD. Prof Otung is a Chartered Engineer with broad and international experience of research and teaching at various universities in Europe and Africa. He founded and continues to manage the popular MSc in Mobile and Satellite Communications at USW.

Jonathan Rodriguez received the M.Sc. degree in electronic and electrical engineering and the Ph.D. degree both from the University of Surrey, U.K., in 1998 and 2004, respectively. He is author of more than 360 scientific works, including eight book titles. In 2005, he became a Researcher at the Instituto de Telecomunicações, Aveiro, Portugal, and in 2008, a Senior Researcher establishing the 4TELL Research Group. He has served as a Project Coordinator for major international research projects, including Eureka-LOOP,FP7-C2POWER, and H2020-SECRET and as a Technical Manager for FP7-COGEU and FP7-SALUS. He is a Chartered Engineer and IET Fellow.