A Lightweight Privacy-Preserving OAuth2-based Protocol for Smart City Mobile Apps

Victor Sucasas, Georgios Mantas, Ayman Radwan, and Jonathan Rodriguez
Instituto de Telecomunicações, Aveiro, Portugal

***Abstract*—In the forthcoming Smart City scenario, users' mobile applications will be of fundamental role towards supporting the envisioned functionalities and services. Mobile users, provided with a smartphone, will be capable of ubiquitously connecting to service providers through their installed mobile applications. However, this connection must be authenticated, which threatens the citizen privacy rights. Privacy-preserving mechanisms have already been proposed in the past; nevertheless, they are based on RSA groups or groups with bilinear pairings, which are inefficient in mobile devices due to its computational complexity. Thus, in this paper, we integrate a lightweight anonymous credential mechanism, suitable for computationally-limited mobile devices, into the user authentication phase of the OAuth2 protocol, which has become a de facto solution for user authentication in mobile applications. The proposed protocol enables citizen's authentication towards service providers, while preserving their privacy. Additionally, the protocol is compliant with the OAuth2 specification, which enables an easy integration in current mobile application implementations.**

I. Introduction

The advances in communications in general, and in wireless communications in particular, created a suitable environment for the emergence of new trend of applications. Emerging applications have moved towards more intelligent applications, with increased number of connected devices, spanning a wide spectrum of objects, ranging from mobile phones, through smart objects, down to wireless sensors. This combination of connected devices has created the perfect environment for the concept of Internet of Things (IoT). Advancing one step further, Smart City concept has emerged with the aim of improving the life quality of citizens by using the Information and Communication Technology (ICT) [1]. Within the Smart City concept, diverse intelligent mobile applications (apps) have emerged as a vital player to enhance and ease the life of citizens [2]. Such applications are usually provided by third-party Service Provider (SP) [3]. Developers usually benefit from the powerful connectivity of mobile devices to create useful mobile apps, which are ubiquitously accessible.

Emerging mobile apps provide a wide spectrum of services, ranging from basic civil services such as accessing/editing your personal information, passing through more private apps such as e-Health, up to very critical apps such as e-banking. All such apps provide enhanced quality of life to citizens, but have access to highly sensitive vital personal information, which require the highest level of security to protect such information from forgery or even worse being hacked and mal-used [4]. Therefore, there is a clear need for resilient authentication and authorization solutions that enable secure access to data with such paramount importance [5]. Nowadays, the most reliable and widely used authentication and authorization protocol for mobile apps to gain access to services hosted by remote service providers is the OAuth protocol. The OAuth protocol is widely adopted by most famous service providers, such Facebook, Microsoft and Google [6], [7]. OAuth protocol was primarily developed to authorize third-party websites to gain access to resources hosted by SP on behalf of end-users. Nonetheless, once OAuth was widely embraced by big industrial names, major service providers (the likes of Facebook and Google) have endorsed it for their user authentication. Additionally, the OAuth protocol was applied to mobile and web applications. In conclusion, OAuth protocol has since become the major authentication and authorization protocol for mobile apps [6]. OAuth2 is the most recent version of OAuth; OAuth1 has since become obsolete [8].

Despite the wide adoption of OAuth2 protocol and the high profile endorsement through the adoption by big names (e.g. Facebook), OAuth2 protocol is still vulnerable to security attacks targeting users' data and credentials [6], [9], [10]. Malicious mobile apps, can be disguised as harmless useful apps; hence installed on by citizens on their mobile device, endangering their data and credentials [11], [12]. Such security attacks do not only threaten citizen's data and information, but also impose risks on their safety, since citizens physical location can be extracted through location information that may be embedded in exchanged messages [5]. These threats still discourage citizens from fully embracing Smart City and its mobile apps. For users/citizens to fully take advantage of the concept of Smart City and enhance their quality of life through its vast apps, resilient security solutions to tackle loopholes in authentication and authorization of citizens in mobile apps have to be developed.

To address this technical challenge, we propose an OAtuh2-based protocol for Smart City mobile apps that addresses the citizens privacy issues, since it allows the users to autheticate towards the authentication servers without revealing identity or account information to the browser installed in the smartphone, other smartphone apps, or to eavesdroppers. It is worth mentioning that our work in [13] already provides a privacy-preserving user authentication mechanism based on OAuth2. However, the proposed approach in [13] is based on Elliptic Curve Cryptography with bilinear pairings, which increases the computational complexity. In this paper however we adopt the credential system proposed in [14] ACL, which does not relay on bilinear pairings, thus it is more convenient to be implemented in low capable mobile devices.

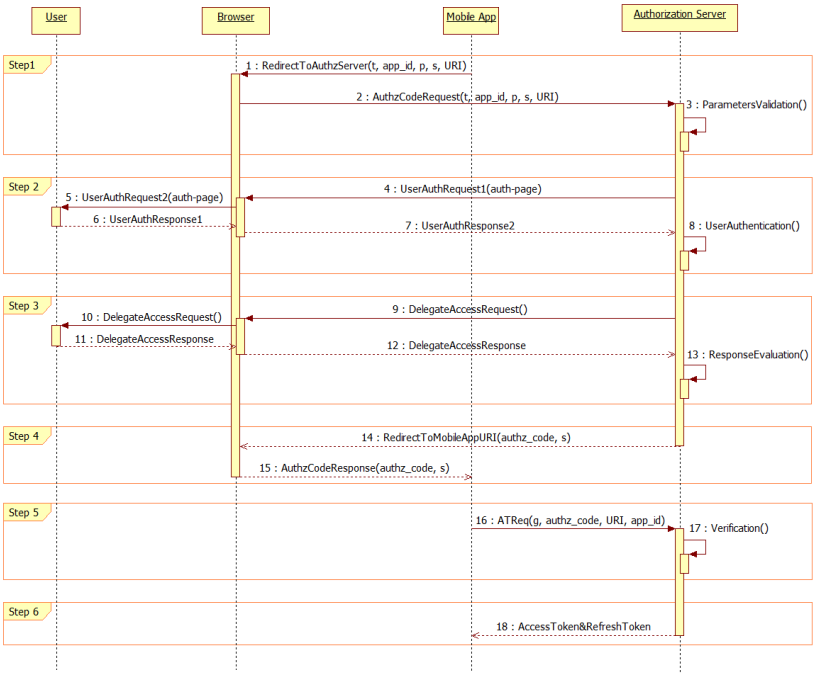

Fig. 1. OAuth2 protocol flow for mobile apps.

The remainder of this paper is organized as follows. Section II presents the OAuth2 protocol and how it is deployed with mobile apps. Section III provides the system model. Section IV details the proposed privacy-preserving OAuth2-based protocol. Section V explains zero-knowledge proofs of knowledge. Section VI provides the security analysis of the proposed protocol. Finally, section VII concludes this paper.

II. OAuth2 and Mobile Apps

The considered OAuth2 protocol defines four entities with different roles [15], as follows: a) Resource Owner: an entity that grants access to its protected resources; b) Resource Server: the server where the resource owners' protected resources are kept; c) the Client: an app requesting access to the protected resources on behalf of the Resource Owner; d) Authentication Server (AS): the server providing access tokens to the Client, after successfully authenticating the Resource Owner and obtaining authorization.

Within the OAuth2 protocol for mobile apps, the Resource Owner is the user who owns protected resources. Additionally, the user installs a mobile app (i.e. the Client) on her/his mobile device. The mobile app is the entity, which requests access to the user's protected resources on behalf of the user. Moreover in the implementations of the OAuth2 protocol for mobile apps, the user agent can be implemented by an embedded web browser or a system native browser. An embedded web browser is a User Interface (UI) component, which constitutes a module of the mobile app and is used to display online contents within the hosting mobile app. On the contrary, the system native browser is the regular browser of the mobile device, which does not constitute part of the mobile app. Both the embedded web browser and the system native browser perform the same role within the OAuth2 protocol flow. For sake of simplicity, we refer to both entities as browser in the rest of the paper [6], [9].

The OAuth2 protocol's implementations are mainly based on browser redirection. The mobile app redirects the browser to the AS, which then interacts with the user, and finally it redirects the browser back to the mobile app. The AS is the entity responsible for performing the authentication of the user. Once authenticated, the mobile app is then identified to the user. The user then has to determine if she/he grants or denies authorization to the mobile app to access her/his protected data. If the mobile app is granted access, the browser is redirected to the mobile app with an authorization code. The mobile app then requests an access token from the AS. The mobile app hence gains access to the user's protected resources hosted in the Resource Server [6], [9]. In more detail, the OAuth2 protocol flow for mobile apps, depicted in Figure 1, consists of 6 steps, as follows [6], [7], [9], [16], [15]:

Step 1: The flow starts by the mobile app redirecting the browser to the AS to request an Authorization Code (msg 1 and 2). More precisely, the mobile app sends a message to the AS via the browser consisting of: i) the response type *(t=code)*; ii) the mobile app identifier *app_id* assigned during the registration process with the AS; iii) the requested permission scope *p*; iv) an optional state parameter *s* to maintain the state between the request and response; and v) a redirection *URI*, which the AS will use to redirect the browser back if access is granted or denied.

Step 2: The AS requests credentials from the user through sending an authentication page (msg 4) to the browser. The user is then prompted by the browser to provide her/his credentials (msg 5). The user provides her/his credentials through the browser, who forwards them to AS (msg 6 & 7) for authentication. The AS then validates the received credentials

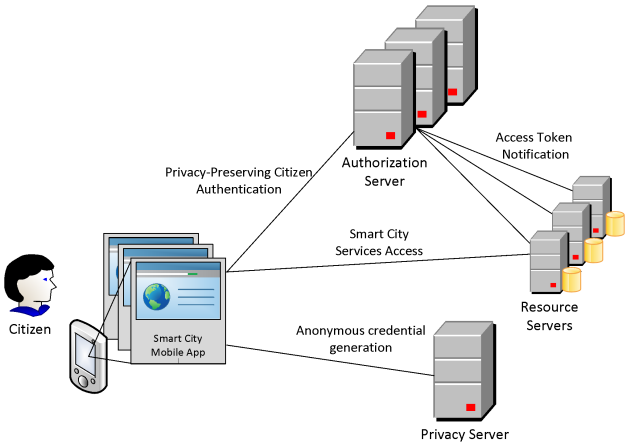

Fig. 2. System model of the proposed privacy-preserving OAuth2-based protocol for Smart City mobile apps.

(msg 8).

Step 3: Once the user is authenticated, the AS proceeds to determine the specific permissions that the user would like to grant to the mobile app (msg 9 & 10). The user then provides her/his preference to the AS, through the browser (msg 11 & 12).

Step 4: After authenticating the user and the permissions are granted, the AS redirects the browser back to the mobile app using the redirection URI. The AS also provides the browser with the Authorization Code authz_code and the state parameter s, provided in Step 1 (msg 14 & 15).

Step 5: The mobile app now can request the access token from the AS. In doing so, the mobile app sends a message with the grant type (g=authorization_code), the Authorization Code authz_code, the redirection URI and the mobile app identifier app_id (msg 16). The AS validates the parameters in the received message (msg 16).

Step 6: The AS provides the mobile app with the requested access token and a refresh token, which is optional (msg 18).

III. OAuth2 Vulnerabilities

The OAuth2 protocol is still vulnerable to attacks by malicious mobile apps, running on the user's mobile device, which can be used to snatch confidential credentials; hence user's privacy can be compromised. If the user is using embedded web browser, a malicious app can compromise the user privacy through injecting JavaScript code which is capable of retrieving user's data once the submit button is clicked [9]. Alternatively, in the case of a system native browser, a malicious mobile app can phish the user's credentials by presenting its own UI instead of allowing the system browser to render the user authentication UI [10].

It is also worth mentioning that encryption is an optional feature in OAuth2, which is frequently neglected by developers. Thus, OAuth2 protocol implementations often involve the transmission of the user credentials and the application ID *app_id* in clear text. Hence, any malicious entity that is able to see the transcript of the OAuth2 communication can profile the user and infer user activities.

IV. System Model

In this paper, we propose a privacy-preserving OAuth2-based protocol for Smart City mobile apps to gain authenticated and authorized access to citizens' personalized services on remote resource servers. The considered system model, described in Fig 2, for the proposed protocol consists of:

- Citizen (i.e. user): A citizen is a subscriber of personalized services and owner of personal data, which are hosted on remote resource servers. In the rest of the paper, citizen and user are used interchangeably.
- Mobile App: It is an app which runs on user's mobile device and provids the citizen with personalized services. The app is responsible for gaining authorization on behalf of the user, i.e. Citizen.
- Privacy App (P-App): The user will be also provided with a privacy-preserving application, P-App, installed in the user's mobile device that will perform the cryptographic operations required in the proposed protocol.
- Resource servers: They are remote servers, usually owned by a third-party, providing personalized services to mobile users.
- Authentication servers (ASs): They are responsible for providing authenticated and authorized access to the smart services hosted on the resource servers.
- Privacy Server (PS): It is a server that issues anonymous credentials to users. PS provides the cryptographic elements that are required for privately authenticating citizens towards ASs.

V. Proposed Privacy-Preserving OAuth2-Based Protocol

This section describes the proposed privacy-preserving OAuth2-based protocol. The proposed protocol achieves user identification and authentication without exposing the user credentials and the user identity towards the browser or other mobile applications installed in the smartphone, which could be controlled by a malicious entity. We integrate a digital anonymous credential system, called Anonymous Credential Light (ACL) [14], into the OAuth2 protocol flow to achieve user privacy. The proposed approach only modifies the information transmitted during the OAuth2 message exchange, but it does not require any modification on the number or the order of the messages transmitted.

The ACL scheme, in which the proposed approach is based on, consists in an efficient anonymous credential system. Its efficiency relies on the fact that it works in elliptic curve groups under the DDH assumption and it does not require bilinear pairings. Our previous approach [13] to provide similar functionality relied on bilinear pairings, which required more computational effort [17], hence increasing the running time in limited mobile devices. Our previous approach provided a pseudonym based system in which users could authenticate towards an AS with different pseudonyms per application. Users were also allowed to renew these pseudonyms on-demand, and off-line, once their time of validity expired, which was a very convenient feature to make the system less dependent from the PS and hence, more autonomous. In the new proposed

protocol this feature is not provided. Users are only given a single credential that is used with different mobile applications, and that requires contact with the PS to be renewed. Users are also required to get one credential per different AS. However, this new approach provides higher efficiency, in terms of computational effort, compared to approaches based on bilinear pairings or RSA that can take several seconds of computation in computationally limited devices.

The proposed approach is constituted by 4 main components: i) *User registration towards the AS and the PS*; The user registers with different AS and creates user accounts, obtaining a unique tag value and a secret value. The user also registers with the PS and obtains a Privacy-Application (P-App). *User anonymous credential generation*; The user contacts the PS and obtains an anonymous credential, which can be used to privately authenticate the user towards the different AS in which the user has created an user account; iii) *Credential verification*; The AS can verify that the user holds a valid credential issued by the PS. iv) *Privacy-preserving user authentication*; it takes place between the user and the AS through the proposed modification of the OAuth2 protocol, which preserves the user privacy from malicious applications or browsers installed in the in the mobile device, or eavesdropper entities accessing the communication between the user and the service provider.

A. User registration with the AS and the PS

The user has to register with an AS, and get a unique tag value m and a secret value S chosen by the user. The user gets one tag and one secret value from each AS, and the AS stores the tag associated with the user identity. The user must access the PS to get an anonymous credential associated with each tag. The user registers with the PS and obtains a privacy application which is installed in the smartphone, P-App. This application is in charge of performing the cryptographic operations required in the proposed protocol. The P-App also stores the tag value m per each AS. The P-App does not store the AS-specific secret values S, which are kept by the user. The P-App also asks the user for $n-1$ user related parameters (such as name, age, etc), $At = (At_1, \ldots, At_{n-1})$. Then the P-App obtains the user-specific parameters $L_i = H(At_i)$ for $i \in [1, n-1]$.

B. User anonymous credential generation

Firstly, the PS performs the setup phase of the ACL system to generate the PS public parameters: It selects a group G of prime order q, and g as generator; Then it selects a hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$. Then it selects $z, h_0, h_1, \ldots, h_n \in G$; The PS picks a secret key $x \in_R \mathbb{Z}_q$ and computes a public key $y = g^x$; Finally, the PS outputs the public parameters $(G, q, g, h_0, \ldots, h_n, y, z)$ and keeps the secret key x.

To generate an anonymous credential, the user, through the P-App, and the PS have to perform a credential issuing protocol, which consists on three steps: registration; preparation; and validation. This process must be performed once per each m value. Hence, users obtain a credential per each AS account.

1) Registration:

- The P-App commits the vector of attributes $L = (L_1, \ldots, L_n)$ where $L_i = H(At_i)$ for $i \in [1, \ldots, n-1]$, and $L_n = H(S)$. To obtain L_n the P-App asks the user to insert his secret value S.
- The P-App chooses a random value $R \in_R \mathbb{Z}_q$, and makes a commitment C, [18], as:

$$C = Commit(L_1, \ldots, L_n, R) = \prod_{i=1}^{n} h_i^{L_i} h_0^R$$

- The P-App sends the commitment C to the PS and perform an interactive standard zero knowledge proof (ZK-proof) [19] to show to the PS that the user knows the opening of the commitment. Note that this ZK-proof is detailed in section VII.

2) Preparation:

- The PS picks a random value $rnd \in \mathbb{Z}_q$, and computes the values $z_1 = Cg^{rnd}$ and $z_2 = z/z_1$. The PS keeps the values z_1 and z_2 and sends to the P-App in the user side the value rnd.
- The P-App receives the value rnd and computes itself the values z_1 and z_2 and picks a blinding factor γ and computes: $\zeta = z^\gamma$, $\zeta_1 = z_1^\gamma$ and $\zeta_2 = \zeta/\zeta_1$, which blinds the values z, z_1 and z_2.
- The P-App also chooses at random a value $\tau \in Z_q$ and computes $\eta = z^\tau$.

3) Validation: The P-App and the PS perform two zero-knowledge proof protocols combined into an OR-proof [19]. This proof consists of the following interactive protocol:

- The PS chooses at random u,c', r_1', and $r_2' \in Z_q$, then computes $a = g^u$, $a_1' = g^{r_1'} z_1^{c'}$, $a_2' = g^{r_2'} z_2^{c'}$. The PS sends to the P-App a, a_1' and a_2'.
- The P-App checks that a, a_1', $a_2' \in G$ hold. Then it picks at random $t_1, t_2, t_3, t_4, t_5 \in Z_q$ and computes: $\alpha = a g^{t_1} y^{t_2}$, $\alpha_1' = a_1'^{\gamma} g^{t_3} \zeta_1^{t_4}$ and $\alpha_2' = a_2'^{\gamma} h^{t_5} \zeta_2^{t_4}$. The P-App computes $\epsilon = H(\zeta, \zeta_1, \alpha, \alpha', \alpha_2', \nu, m)$, where m is the tag received from the AS. Then the P-App sends to the PS the value $e = \epsilon - t_2 - t_4 \ mod \ q$.
- The PS computes $c = e - c' \ mod \ q$ and $r = u - cx \ mod \ q$ (note that PS is using its secret key x). Then PS sends to the P-App in the user side the values $(c, r, c', r' = \{r_1', r_2'\})$.
- The P-App receives the values from the PS and computes: $\rho = r + t_1 \ mod \ q$, $\omega = c + t_2 \ mod \ q$, $\rho_1' = \gamma r_1' + t_3 \ mod \ q$, $\rho_2' = \gamma r_2' + t_5 \ mod \ q$, $\omega' = c' + t_4 \ mod \ q$, $\mu = \tau - \omega' \gamma \ mod \ q$.

Finally, the P-App obtains the blinded signature: $\sigma = (m, \zeta, \zeta_1, \rho, \omega, \rho', \omega', \mu)$, where $\rho' = \{\rho_1', \rho_2'\}$. The P-App stores the tuple (σ, m, ζ_1), and also the factors γ and rnd. The tuple (σ, m, ζ_1) is compose by three values: the blinded signature σ signed by the PS, the tag value m associating the user to a certain AS, and the blinded commitment ζ_1 computed with the users attributes.

It is important to mention that the PS cannot link the showing of this signature to its issuing. The PS cannot recognize signature variables, nor the blinded commitment ζ_1,

which includes the commitment C, neither the tag m that has been blindly signed. Hence the PS cannot link a user showing this credential to a previous credential issuing procedure. This feature provides users with an additional privacy feature, since the PS cannot track users activities. This feature differs from our previous proposed scheme provided in [13].

4) Signature verification: A verifier can now that the tuple (σ, m, ζ_1) was correctly constructed if the following equation holds and $\zeta_1 \neq 1$:

$$\omega + \omega' = H(\zeta, \zeta_1, g^{\rho} y^{\omega}, g^{\rho'_1} \zeta_1^{\omega'}, h^{\rho'_2} \zeta_2^{\omega'}, z^{\mu} \zeta^{\omega'}, m) \; mod \; q0$$

C. Credential verification

A user that has performed an account registration with an AS, obtaining a secret value S and a tag value m, and that has performed a credential issuing process with the PS, will have stored in its P-App a tag m, a signature σ, and the blinded commitment ζ_1. The P-App also stores the values (rnd, γ). The user can show this tag m and the signature σ to a verifier, which in our proposed protocol will be the AS. The AS can verify the validity of such signature on the tag m, and identify the user associated with the unique tag m. The AS can be sure that the user knows the secret parameter S with which the blinded commitment ζ_1 was constructed without requiring the user or the P-App to send or disclose this value, by means of a zero knowledge proof.

To proof the identity of the user, the P-App prompts the user to insert its S value associated with the respective AS and compute the attribute $L_n = H(S)$. The P-App also obtains the committed values $L_1, \ldots, L_{n-1}$, computed from the stored attributes $At_1, \ldots, At_{n-1}$, i.e. $L_i = H(At_i)$. Then the P-App performs an interactive zero-knowledge proof of knowledge of the opening of the blinded commitment ζ_1, with the verifier, to show the knowledge of the attributes $L_i, \ldots, L_n$, where L_n is also known by the AS. This ZK-proof is detailed in the following section.

D. Privacy-preserving user authentication

The proposed OAuth2-based procotol integrates the credential verification algorithm, which includes the ZK-proof of a blinded commitment, into the OAuth2 protocol flow. The credential verification is performed between the P-App and the AS, the user is only prompted by the P-App to insert the secret value S. In the proposed protocol, the OAuth2 procotol is performed according the description of the original OAuth2 specification, which has been detailed in section II, but integrating the information required for the credential verification. We detail below the additional information included in some of the OAuth2 messages, namely the messages 4 to 13:

4.UserAuthRequest1() The AS sends an authentication request to the user, which is managed by the browser.

5.UserAuthRequest2() The browser initiates the P-App which will be the user interface. The P-App prompts the user to include the secret for the AS, i.e. the value S. Then it includes $L_n = H(S)$ in the list of secret values $(L_1, \ldots, L_n, R)$.

6.UserAuthReponse1() The P-App computes the value t and θ of the ZK-proof, see section VII. The P-App sends the tuple $(\sigma, m, \zeta 1)$ that includes the signature, the tag and the blinded commitment. The P-App also send the values t and θ of the ZK-proof.

7.UserAuthResponse2() The browser forwards to the AS these values.

8.UserAuthentication() The AS looks up the tag m to identify the user, and the associated secret S. Then the AS picks a random value $c \in \mathbb{Z}_q$ to continue with the interactive ZK-proof. The AS performs the signature verification algorithm to check the validity of the signature.

9.DelegatesAccessRequest() The AS sends c to the P-App.

11.DelegatesAccessResponse() The P-App computes the values s_i for $i \in [0, n-1]$ and ξ and sends it to the AS to complete the zero knowledge proof.

13.ResponseEvaluation() The AS accepts or rejects the ZK-proof. In case of acceptance the AS continues the OAuth2 protocol, otherwise it halts and sends the browser a failure notification.

VI. Security analysis

The correctness and the security of the anonymous credential system used in our privacy-preserving OAuth2-based protocol is already demonstrated in [14]. The identity privacy feature of the proposed scheme relies on the fact that the identity of the user is never disclosed or transmitted. User get identified and authenticated though the transmission of the AS-specific unique tag m. Although the tag and the signature are transited in cleartext, the knowledge of the tag and the signature does not enable an attacker to impersonate the user, since the knowledge of the secret values is required to succeed in the ZK-proof performed during signature verification. Moreover, one of the secret values is kept by the user and never stored by the P-App, hindering the leakage of sensitive information from the P-App. It is also worth mentioning that the blinded signature scheme ensures that the PS cannot link the show of a signature to its issuing, hence the PS cannot track users activities.

VII. Zero-knowledge proofs of knowledge

A zero-knowledge proof (ZK-proof) [19] is an interactive protocol between two parties in which a Prover wants to proof to a Verifier that it holds a secret, but without revealing anything more than the fact that it knows that secret. In our protocol, the Prover is the User/P-App and the Verfier is the PS and the AS during the credential issuing and signature verification respectively.

A. Interactive ZK-proof of the openning of a commitment

The signature issuing process requires a standard ZK-proof during the registration phase of the anonymous credential generation. In a standard ZK-proof of a representation, the Prover knows $C = \prod_{i=1}^{n} h_i^{L_i} h_0^R$, the public parameters $(h_0, \ldots, h_n)$, and the secret values $(L_1, \ldots, L_n, R)$. The Verifier knows C and the public parameters. Both the Prover and the Verifier perform the following interactive protocol, which ends with the

rejection or acceptance of the proof by the Verifier depending on whether the last equation holds.

$Prover$		$Verifier$
$(L_1, \ldots, L_n, R); C = \prod_{i=1}^{n} h_i^{L_i} h_0^R;$		C
$(h_0, \ldots, h_n)$		$(h_0, \ldots, h_n)$
$r_i \in \mathbb{Z}_q; i \in [0, n];$		
$t = \prod_{i=0}^{n} h_i^{r_i};$		
	$\xrightarrow{t}$	
		$c \in_R Z_q$
	$\xleftarrow{c}$	
$s_i = r_i - cL_i; i \in [1, n]$		
$s_0 = r_0 - cR;$		
	$\xrightarrow{s_0, \ldots, s_n}$	
		$\left(\prod_{i=0}^{n} h_0^{s_0}\right) C^c \stackrel{?}{\equiv} t$

B. Interactive ZK-proof of a blinded commitment, with the knowledge of one secret value

The verification of the signature requires a ZK-proof on the values used for the blinded commitment $\zeta_1 = z_1^{\gamma}$ where $z_1 = Cg^{rnd}$, and C is the commitment $C = \prod_{i=1}^{n} h_i^{L_i} h_0^R$. In this proof the Verifier knows ζ_1, one of the secret values $L_n = H(S)$ and the Prover knows the secret values $(L_1, \ldots, Ln, R)$, the blinding factor γ, and the random factor rnd. Both parties know the public parameters $(h_0, \ldots, h_n)$. The objective of the ZK-proof is that the Verifier is convinced that the Prover knows the secret values with which the blinded commitment ζ_1 was constructed. Both Prover and Verifier perform the following interactive protocol, which ends with the rejection or acceptance of the proof by the Verifier depending on whether the last equation holds.

$Prover$		$Verifier$
$(L_1, \ldots, L_n, R);$		ζ_1, L_1
$(h_0, \ldots, h_n); \zeta_1; rnd; \gamma;$		$(h_0, \ldots, h_n)$
$r_i \in \mathbb{Z}_q; i \in [0, n-1];$		
$t = \prod_{i=0}^{n-1} h_i^{r_i};$		
$\theta = g^{rnd\gamma}$		
	$\xrightarrow{t, \theta}$	
		$c \in_R \mathbb{Z}_q$
	$\xleftarrow{c}$	
$s_i = r_i - cL_i\gamma;$		
$i \in [1, n-1];$		
$s_0 = r_0 - cR\gamma;$		
$\xi = h_n^{c\gamma}$	$\xrightarrow{s_0, \ldots, s_{n-1}, \xi}$	
		$\left(\prod_{i=0}^{n-1} h_i^{s_i}\right) \xi^{-L_1} \zeta_1^c \stackrel{?}{\equiv} t\theta^c$

VIII. CONCLUSION

This paper proposes a lightweight privacy-preserving OAuth2-based protocol for smart city mobile apps. The proposed scheme enables users to obtain credentials to identify and authenticate themselves, towards authentication servers, without disclosing their identity to malicious entities located in the user's mobile device or eavesdropping the communication between the user and the authentication server. Moreover, the privacy server is in charge of issuing the anonymous credentials and cannot link the issuing of a credential to the usage of such credential and hence, enforcing users privacy. The proposed scheme is integrated into the OAuth2 protocol, without modifying the original message flow. Although the proposed scheme cannot provide an autonomous pseudonym self-generation scheme like the approach in [13], it is based on a more lightweight signature scheme, hence more suitable for hardware limited mobile devices.

REFERENCES

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

[2] Solanas et al. Smart health: A context-aware health paradigm within smart cities. *Communications Magazine, IEEE*, 52(8):74–81, Aug 2014.

[3] 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.

[4] Catalin Gosman, Tudor Cornea, Ciprian Dobre, Florin Pop, and Aniello Castiglione. *Putting the User in Control of the Intelligent Transportation System*, pages 231–246. Springer International Publishing, Cham, 2016.

[5] Antoni Martinez-Balleste, Pablo Perez-Martinez, and Agusti Solanas. The pursuit of citizens' privacy: A privacy-aware smart city is possible. *IEEE Communications Magazine*, 51:136–141, 2013.

[6] Eric Y. Chen, Yutong Pei, Shuo Chen, Yuan Tian, Robert Kotcher, and Patrick Tague. Oauth demystified for mobile application developers. In *2014 ACM SIGSAC*, CCS '14. ACM, 2014.

[7] San-Tsai Sun and Konstantin Beznosov. The devil is in the (implementation) details: An empirical analysis of oauth sso systems. Aug 2012.

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

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

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

[11] 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.

[12] 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.

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

[14] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In *Proceedings of the 2013 ACM SIGSAC*, CCS '13, pages 1087–1098, New York, NY, USA, 2013. ACM.

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

[16] G. Developers. Using oauth 2.0 for installed applications. In *Available: https://developers.google.com/identity/protocols/OAuth2InstalledApp*, 2015.

[17] Patrik Bichsel, Jan Camenisch, Thomas Groß, and Victor Shoup. Anonymous credentials on a standard java card. CCS '09, pages 600–610, New York, NY, USA, 2009. ACM.

[18] Ivan Damgård. *Lectures on Data Security: Modern Cryptology in Theory and Practice*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[19] Ivan Damgård. *Lectures: http://www.cs.au.dk/ ivan/Sigma.pdf*. 2010.