To appear in the Proceedings of The 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-13)

Portable Personal Identity Provider in Mobile Phones

Md. Sadek Ferdous
School of Computing Science
University of Glasgow
Glasgow, Scotland
Email: m.ferdous.1@research.gla.ac.uk

Ron Poet
School of Computing Science
University of Glasgow
Glasgow, Scotland
Email: ron@dcs.gla.ac.uk

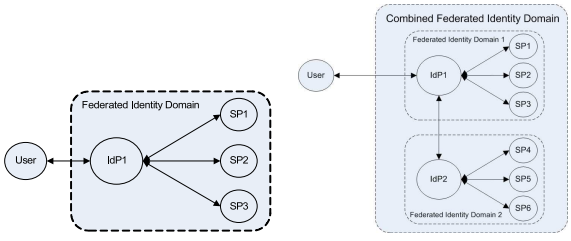
***Abstract*—This paper analyses the prospect of having a Portable Personal Identity Provider (PPIdP, in short) in the mobile phone. The ubiquitous presence of powerful mobile phones equipped with high speed networks can be utilised to make the mobile phone act as a portable and personal Identity Provider (IdP, in short) on behalf of their users. Such an IdP would be helpful for the user in the sense that it will provide a central location to manage different user attributes which are generally scattered among different service providers in the traditional setting of online services. In addition, the user needs to trust the provider to store those attributes securely which may not be always honoured and crucial user attributes may be abused. Creating a Personal Identity Federation using a personal IdP can tackle many of these stated problems. Moreover, such an IdP may provide additional advantages. We have developed such a Mobile IdP for the Android platform based on the Security Assertion Markup Language (SAML) and OpenID as a proof of concept using the Jetty Web Server. In this paper, we discuss the functionalities of our developed IdP and the technical challenges we have faced. Moreover, we analyse the security, privacy and trust issues involved in having such an IdP and the advantages it offers.**

***Keywords*-Identity Management, Identity Provider, Federated Identity Management, Security Assertion Markup Language, OpenID, Trust.**

I. Introduction

With the continuous evolution of web-enabled (or online) services in the last decade or so, the way those services can be accessed has changed considerably. As the number of such online services as well as the user-base was expanding rapidly, the management of user identities became challenging, both for service providers and for users. Identity Management (IdM, in short) was introduced by the industry to facilitate online management of user identities which resulted in various different Identity Management Systems (IMS, in short). Formally, Identity Management consists of technologies and policies for representing and recognising entities using digital identifiers within a specific context [1]. Shibboleth [2], OpenID [3], Microsoft's CardSpace [4], etc. are all examples of different IMS.

Each IMS usually has three different actors: *Identity Provider (IdP)* - an entity responsible for managing digital identities of users and providing identity related services to different Service Providers; *Service Provider (SP)* - an entity to provide web-enabled services to the users based on the identity information (identifiers and/or attributes) received from the IdP and *User (Client)* - an entity that receives services from a SP. Among different IMS, Federated Identity Management (FIM, in short) System has gained considerable attention. It is based on the concept of Identity Federation (also known as Federated Identities or Federation of Identities). In the ITU-T X.1250 recommendation, a federation is defined simply as "An association of users, service providers and identity providers" [5]. In other words, a federation with respect to Identity Management is a business model in which a group of two or more trusted parties legally bind themselves with a business and technical contract [6], [7]. It allows a user to access restricted resources seamlessly and securely from other partners from different Identity Domains. An identity domain is the virtual boundary, context or environment in which a digital identifier is valid [7]. Single Sign On (SSO) is the capability that allows users to log in one system and then access other related but autonomous systems, from the same or different identity domains, without further logins.



(a) Type 1. (b) Type 2.

Figure 1. Federated Identity Domain.

A federated identity domain can be formed consisting of only one IdP within an identity domain and more than one SP with each SP residing in a separate identity domain (Type 1 in Figure 1). Several federated identity domains can be combined to form a larger federated identity domain where each smaller federated domain is of Type 1 (Type 2 in Figure 1). The issue of trust is a fundamental concept in FIM as different autonomous bodies need to trust each other inside the federation. Such parties inside a federation is said to

form the so-called Circle of Trust (CoT).

Most IMSs, including the FIM, have been designed to help organisations to manage their user identities whereas the management of user identities by the user is mostly overlooked [8]. This results in serious negative consequences. Firstly, the user has limited control once the attributes are stored in the IdP and has no way of knowing how these attributes are handled at the IdP. Secondly, the management of such attributes becomes increasingly difficult for the user once the number of IdPs starts increasing. The proposed portable personal IdP can solve both problems.

Current mobile phones are equipped with powerful hardware, intuitive software and advanced capabilities and are rightfully called Smartphones. The ubiquitous presence of such powerful mobile phones equipped with high speed mobile networks enables users to access online services literally everywhere. It makes them the suitable choice as the host of the poratble personal IdP which we introduce in this paper. The contributions of our paper are:

- Being a novel term, we, at first, provide a concrete definition of the term "Portable Personal Identity Provider (PPIdP)" and "Personal Identity Federation".
- We introduce the notion of *Personal Attribute Store (PAS)* for mobile phones which can be used to store crucial user attributes securely and will be used by the PPIdP during a service access scenario.
- We describe our developed proof of concept of the PAS and the PPIdP based on the Security Assertion Markup Language (SAML) [9] and OpenID [3] to illustrate the applicability of our approach. Note that there are other IdM Protocols available such as WS-Federation ([10]) and OAuth [11], but in this paper, our main focus will be on SAML and OpenID.
- Since SAML was not envisioned to be applied in such a setting, many established trust assumptions do not hold. Our approach requires formulating novel trust assumptions. We analyse these trust issues in details while describing our proof of concept.
- We also analyse different security and privacy issues involved with our approach, the advantages it offers and the limitations it has.

The reminder of this paper is organised as follows. We discuss a few key works related to this topic and analyse their strengths and weaknesses in Section II. The concept of PPIdP and related issues are defined in Section III. The notion of the PAS is introduced and our developed proof of concept of the attribute store and the PPIdP are discussed in Section IV. Different security and privacy issues are analysed in Section V and a discussion of advantages and limitations along with the extensibility and future work of our approach can be found in Section VI. We conclude in Section VII.

II. Related Work

Each mobile phone is equipped with a Subscriber Identification (or Identity) Module (SIM) which stores a unique International Mobile Subscriber Identity (IMSI) number and the related cryptographic keys that are used to uniquely identify and authenticate the user and then register the SIM into the respective mobile network [12]. In such, the mobile operator can be used as the IdP using the SIM as the facilitator and is commonly known as the "Mobile Identity Management" [13]. Several standards such as Generic Authentication Architecture (GAA) [14] and Generic Bootstrapping Architecture (GBA) [15] have been put forward as part of the 3rd Generation Partnership Project (3GPP) [16] to integrate such mobile identity management with the existing web services. These standards allow to leverage the strong authentication mechanism using SIM to access services. The problem with this approach is that the IdP (the mobile operator) controls all user attributes and the user has no or limited control over the attributes. Moreover, users often change mobile networks and/or the SIM meaning that attributes stored with the previous operator may not be used to access web services when the user has changed the mobile network. For these reasons, we will not explore this approach any further.

Being a novel topic, there have been relatively fewer works related to the theme of this paper. The initial concept of a Personal Identity Provider can be found in [17], in which the authors proposed their idea of *Identity-aware Devices*. An identity-aware device contains a local IdP hosted inside a Trusted Platform Module (TPM) in the device. The local IdP is coupled with a telecom provider which acts as the principal IdP. Our approach is similar to this approach with one major difference: their approach requires the user to rely heavily on the principal IdP whereas we want to decouple the reliance on any other external IdPs. Another work related to our theme can be found in [18], in which a portable Liberty Alliance[1]-enabled IdP installed in the mobile phone is presented. The IdP is accompanied by a Relay Server which is maintained by the Mobile Operator. However, this paper does not discuss how the federation can be established nor does it describe how user attributes will be transmitted to the SP. In [20], the author proposes a type of device-centric identity which allows a user to authorise a device to use a cryptographic key to identify the user and then to allow accesses to online services and user attributes. However, it is not outlined how such a proposal can be realised in practice.

Current popular Smartphone Operating Systems such as Android, iOS, Windows, etc. are associated with their respective IdPs. For Android, the IdP is Google Account (accounts.google.com); whereas it is AppleID (https://appleid.apple.com/) for iOS and Windows Live ID

[1]More precisely the "Identity Federation Framework" (ID-FF) formulated by the [19] Liberty Alliance (LA, in short) consortium.

(www.live.com) for Windows. Among them, Apple ID and Windows Live ID are mostly used to access services associated with Apple and Windows respectively. Only Google provides a Federated Login service based on the OpenID and OAuth protocols to allow users to use their Google accounts to access services outside the Google identity domain [21]. Even though it adds an element of portability, the Google IdP acts just like any third party IdP holding user attributes and so cannot be tagged as a Personal IdP.

III. Portable Personal Identity Provider

At first we define the term Portable Personal Identity Provider and then we utilise this term to define the concept of Personal Identity Federation.

Definition: *Portable Personal Identity Provider (PPIdP)* - A type of Identity Provider that is under the full control of a user and resides in a mobile device owned and/or used by the user can be defined as the Personal Identity Provider. It is the user who must decide what attributes should be stored in such an IdP and which attributes should be released to which SP by this IdP.

Since the user can add/edit/update attributes into this IdP, the IdP must provide an interface to do so as well as an interface to set any attribute release policies. It is also the responsibility of the IdP to ensure that all user attributes are stored safely and securely. In addition, such an IdP adds the feature of portability which allows the user to use it while on the move. In the context of this paper, our focus is mainly on the mobile phone, however, the concept can easily be adapted to other mobile devices as well. Windows CardSpace, a former initiative from Microsoft, could act as a type of personal IdP [4] along with other functionalities and was included with several versions of Windows operating system such as Windows 7 and Windows Vista. However, it was not portable since it was only available for desktops. Currently there is no such personal yet portable IdP and our implementation described here aims to fill in this gap.

Definition: *Personal Identity Federation* - When an Identity Federation is created using a PPIdP, it can be defined as the Personal Identity Federation.

In a way, this is an extension of the traditional Identity Federation and also can be of two types: Type 1 & Type 2.

A. Necessity

With the advent of Identity 2.0, User-centric Identity Management has gained considerable attention. There is no formal definition for the term User-centric Identity Management, however, it is believed to mean an IMS which allows users to have considerable control over their attributes so that users can choose which attributes they want to release to a SP [22]. Windows CardSpace and OpenID are examples of the User-centric IMS. However, before the attributes can be released, users must register to the respective IdP and provide those attributes which are then stored in the IdP. With some recent incidents in which sensitive user data (read attributes) were stolen from some major global companies (which also acts as, more or less, SILO IdPs for their respective services [1]) such as Sony (77 millions of user accounts were affected [23]), LinkedIn (more than six million passwords had been stolen [24]), Apple (over 1 million Apple IDs fell into the wrong hand[25]), users are rightfully worried to store crucial user data into these IdPs. Moreover, in many IMS, a legal contract between the IdP and the SP dictates the handling of user attributes between themselves. However, there may not be any legal contract between the user and the IdP and therefore the handling of attributes may be governed by the respective Terms & Conditions (T&C). The absence of any legal contract between the user and the IdP means that the handling of user attributes is only bound by a trust assumption where the user can only hope that the respective party will honour the imposed trust [26].

Furthermore, when there are many more IdPs, users will feel overloaded to manage attributes spread over these IdPs and also the same attributes will be stored in multiple places resulting in unnecessary redundancy and thus increasing the chance of theft. All these problems can be tackled significantly by harnessing the true potential of the User-centric IdM by letting users not only choose which attributes to release but also to store them in fewer IdPs which are preferably under their own control. A personal IdP is an excellent choice for this since it allows the user to manage all crucial attributes from a central place. We understand that sometimes it is required by the SP to have higher assurances about user attributes when the user tries to access sensitive services such as Governmental or Health Services. In such scenarios, the SP would only trust attributes released by a highly trusted IdP. The number of such sensitive online services will be essentially low compared to the number of general non-sensitive services. This means that the number of highly trusted IdPs will be much lower than the number of general IdPs since there is a significantly higher administrative cost associated with establishing and maintaining a highly trusted IdP. In fact, not all IdPs need to be a highly trusted IdP. In most of these general IdPs, user attributes are mostly provided by the users themselves. Our main focus is to get rid of all these general IdPs and replace them with the PPIdP. This will allow users to maintain a lower number of highly trusted IdPs that could be used to access only those services that ask for attributes to be released from a highly trusted IdP and a single PPIdP which could be used for the rest of other general services.

In addition, such an IdP has added advantages. The traditional IdPs cannot produce many dynamic attributes such as the current location data of the user. We can use

the PPIdP to create dynamic attributes whenever required and provide them to SP during any service access scenario.

B. Key Issues

Designing and developing a PPIdP requires several key issues to be addressed which are discussed below:

- *Technical Challenges*: The existing APIs of different IdM protocols (e.g. SAML and OpenID) have not been developed with any locally hosted IdP in mind. The general assumption is that both the IdP and the SP are online and visible to each other. Since our PPIdP will be essentially hosted inside the mobile device, this assumption does not hold. In addition, all IdM protocols are based on standard web technologies, therefore, we need to ensure that the IdP is compatible with such web technologies.
- *Trust Issues*: Implicit in every Identity Management is the issue of trust. Hence, we need to closely examine the trust assumptions and requirements that might arise while using our IdP.
- *Attribute Storage*: Every IdP has to store and handle user attributes. The IdP must provide an interface to allow the user to provide new attributes and update existing ones. Also the IdP should ensure that these attributes are stored safely and securely.
- *Attribute Authorisation*: In addition, the IdP must provide an interface to allow the user to set up any Attribute Release Policy (ARP). This will ensure several key requirements of an IMS such as Explicit Consent, Transparency and Data Control [27].

IV. Proof of Concept

To prove the applicability of our proposed PPIdP, we have developed a proof of concept prototype. The current implementation of the prototype is based on SAML and OpenID protocols. At first, we need to choose a platform from several popular platforms namely Android, iOS, Windows and Blackberry. Since both SAML and OpenID have a Java API, we have been looking for a platform with strong support for Java. With this in mind, Android has been chosen as our development platform. The architecture of the prototype is given in Figure 2: The prototype consists of two major components called the Personal Attribute Store (PAS) and the IdP Component. Both of them are discussed below.

A. Personal Attribute Store

As the name suggests, the Personal Attribute Store is where all user attributes will be stored and is a necessary companion of the proposed IdP. It consists of a back-end database to store the attributes and a user interface for users to manipulate their attributes. It also has interfaces to communicate with internal sensors (such as the GPS receiver) of mobile phones to collect contextual information and store them as dynamic attributes. Android is also bundled with

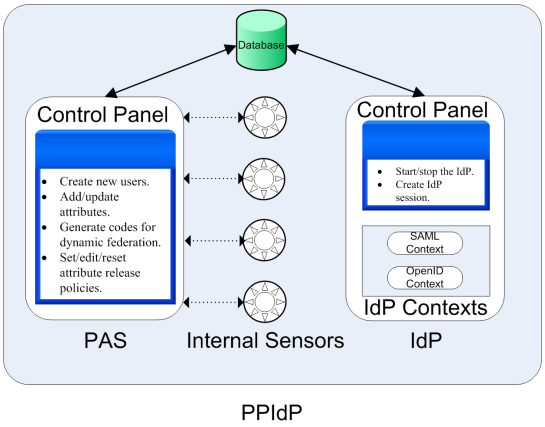


Figure 2. Architecture of the PPIdP Prototype.

support for the SQLite Database [28] to be used as the back-end database to store user attributes. To ensure the security of stored attributes, it is preferred that attributes are encrypted before storage. We have used an open source library called SQLCipher [29] to encrypt attributes before storage and decrypt during usage. SQLCipher provides fast and secure 256-bit AES encryption of SQLite database files and that alleviates us from the manual encryption and decryption of attributes.

The PAS has been developed as an Android app. Every first-time user of the PPIdP must create an account by providing a username and password along with other attributes. The user must use that identifier (and the credential) to manipulate attributes and also to start the IdP. Once the user is logged in, a session is created and she is presented with the main control panel of PAS which has several options (Figure 3). The user can choose the *Add* option to add a new attribute, *Edit* and *Delete* options to edit and delete any existing attribute(s) respectively, *Exit* option to exit the AS and *Sign Out* option to sign out by deleting the existing session. If the user exits the AS without signing out, the session will be kept for an hour. When the user launches the AS next time and the session has not expired, the user will be automatically taken to the main control panel of the PAS. This saves the user to log in every time the AS is launched. The panel also shows the already added attributes, if any, at the bottom part of the screen. The functionalities of other options will be discussed in due course. The user also needs to generate a secret key that will be used for encryption/decryption of user attributes into the database. The secret key is generated as a configuration parameter by hashing two values that the user needs to provide: the mobile phone number of the user and a chosen password.

B. Identity Provider Component

The IdP component is responsible for providing IdP services and also has been developed as an Android app. The IdP has two sub-components: the IdP Context and the

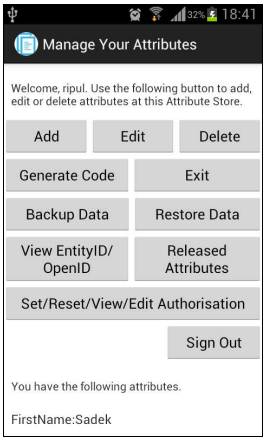


Figure 3. Main control panel of the PAS.

Figure 4. Main control panel of the IdP.

Control Panel. The IdP context actually consists of several servlets. The SAML servlet is responsible for trapping the SAML requests and then respond to them. The OpenID context is to handle OpenID requests. To develop the SAML servlets, at first we have considered using the OpenSAML library [30]. OpenSAML is a set of open source Java and C++ libraries that can be used for SAML development. However, it has quite a large number of external dependencies (more than 20 external jar files) which would increase the size of the app considerably and many of these jar files could not be compiled with the Dalvik VM of the Android platform. To overcome the problem, we developed a subset of the SAML protocol based on the Web Browser SSO Profile and HTTP Post binding. In this way, we have managed to reduce the size of the app considerably. The OpenID servlet has been developed using the OpenID4Java Libraries [31].

The user can use the same username/password pair of the PAS to log in to the IdP component and to access the control panel. The control panel provides the option to Start/Stop the IdP component using the *Start/Stop* button respectively (Figure 4). When the *Start* button is touched, the IdP component starts running in the background. Then the user can use their preferred browser of the mobile phone to access federated services. When an authentication request arrives at one of the contexts, it can access the same back-end database mentioned previously to retrieve user attributes and then can create responses using those attributes. We have used the simpleSAMLphp [32] and the JanRain PHP OpenID libraries [33] for deploying SAML and OpenID service providers respectively. At this point, we will discuss some key challenges we have faced to develop our PPIdP.

1) Web Server: The first challenge was to find a suitable web container for the Android platform. After a thorough search, Jetty Web Server, an open source HTTP Server and Java Servlet Container [34], seems to be the only candidate that met our requirements.

2) Visibility Issues: The second challenge was to solve the problem of visibility between the IdP and SP. In the traditional setting, both IdP and SP are online and visible to each other. In our case, the IdP is installed in the device as a local web server and therefore is not visible to the SP. We have solved this problem by embedding an XHTML Form submitted to our IdP using JavaScript. Since JavaScript runs into the browser, it is possible to contact the respective IdP context using the *localhost* URL. This is a standard technique used in the SAML SSO Browser profile. We have just adopted the same behaviour and modified the code of the simpleSAMLphp and JanRain PHP libraries for our scenarios. Note that the IdP has no problem contacting the SP since the SP has be to online to provide online services.

3) SAML Trust Issues & Dynamic Federation: The third challenge was to handle the fundamental trust issues involved in the SAML. A SAML SP needs to trust that the IdP will authenticate the user using appropriate security mechanisms and release attributes to the SP as per the contractual agreement [26]. Similarly, the IdP has to trust that the SP will not abuse the released attributes and use them only for the stated purpose as per the agreement. Trust in SAML is initiated when the respective IdP and the SP exchange metadata between themselves. The exchanged metadata is then stored in a repository which is used to update the respective Trust Anchor List (TAL) [35].The SP and IdP will only trust entities which can be found in the respective TAL.

Metadata must be exchanged before the IdP and the SP can interact with each other and the exchange takes place externally during/after the parties bind themselves with a technical contract. Now exchanging metadata in this way has two potential problems. Firstly, pre-configuring trust before any interaction hinders the establishment of a federation between two prior unknown parties in a dynamic fashion. Secondly, pre-configuring trust between two organisations by exchanging metadata is usually carried by the respective administrators. Now, we are faced with a dilemma with our case of PPIdP. We do not want to burden users with the task of metadata exchange and at the same time there must be a way for SAML-enabled parties to exchange metadata with each other. We can address both these issues by allowing users to create federations in a dynamic fashion.

The authors in [35] presented a mechanism by which a SAML federation can be created in a dynamic fashion. There is no administrative overhead associated with the technique allowing any general user to establish federations whenever required. We have utilised the same approach for creating a Dynamic Federation. According to that method, the entity ID (a unique identifier given to an IdP or a SP) of each SAML entity (IdPs and SPs) will represent the endpoints from where their respective metadata can be fetched dynamically. At first, the user will need to generate a temporary code from the IdP. The SP is assumed to provide an interface

to add the IdP, preferably at the Where Are You From (WAYF) of the SP. Then the user will input the entity ID of the IdP into that page along with the code. The SP will send a request for dynamic federation into that entity ID along with the respective code and the SP's entity ID. After validating the code, the IdP will use the SP's entity ID to fetch SP's metadata and save it to the IdP's TAL. Then the IdP will send back its metadata to the SP which will be ultimately stored into the SP's TAL. Since the SP might not trust any dynamically added IdP, any such IdP will be treated as an untrusted entity to the SP whereas the IdP will treat the SP as an untrusted entity initially. The SP will be promoted as a semitrusted entity only when a user releases a subset of attributes to that SP. The same mechanism can be used to federate two IdPs by a user of both IdPs and thus enabling a Type 2 federation. In this case, one of IdPs is assumed to be fully trusted to the SP by having a formal trust agreement between them. This will allow the fully trusted IdP to delegate the authentication task to the dynamically added IdP. The dynamic IdP will release user attributes to the trusted IdP which, then, will release them to the SP. To avoid any privilege escalation, the authors proposed to utilise the NIST LoA (Level of Assurance or Level of Authentication [36]) level of 1 to signify that the level of assurance is low since attributes are from the PPIdP.

C. SAML Protocol Flow

In our work, we have adopted a similar approach, as discussed above, for creating a dynamic federation. A SAML IdP and a SAML SP have been deployed using simple-SAMLphp. Their metadata have been exchanged beforehand to ensure that are considered fully trusted to each other. Now to create a Type 1 dynamic federation, the user first generates a temporary code using the user interface of the PAS. Then the user visits the SP where the user is presented with a form to allow her to create the dynamic federation. The chosen entity ID for our IdP is *https://localhost:8443/saml/idp*. The user inputs that entity ID along with the code. Then the previously described mechanism for exchanging metadata takes place. Note that, the SP must use the JavaScript technique as described before to contact with the IdP. At the end, the respective metadata will be exchanged making the PPIdP and the SP a part of the Personal Identity Federation. At this point, the usual SAML Protocol flow takes place. The user visits the SP to access one of its resources. The SP forwards the user to the WAYF Page where she chooses the PPIdP. The SP creates a SAML authentication request and forwards the request to the PPIdP using an embedded XHTML form and submitting the form automatically using JavaScript. The SAML context of the IdP traps the request and presents a login form to the user. The user logs in using her username/password. Once the authentication is successful, a new form with all user attributes are presented and the user selects the attributes that she wants to release to the SP. A SAML assertion using the selected attributes is created and returned to the SP. The SP retrieves the user attributes and makes an authorisation decision to allow or reject the user to access the resources (Figure 5). The PAS also allows the option to create a pre-configured Attribute Release Policy for any specific SP using the *Set/Reset/View/Edit Authorisation* option of the PAS main control panel (Figure 3). The IdP will use the ARP to filter out any attributes before the attribute selection process.

Since the PPIdP has been dynamically added to the SP, the SP must treat it as an untrusted entity even if the IdP acts honestly. In this way, the SAML implementation of the IdP will essentially act equivalently to the OpenID. Thus this technique will be suitable for general SPs which do not require users attributes coming from a highly trusted IdP. Interestingly, some attributes (such as credit card information) are self-certifiable. It does not matter if they are coming from a highly trusted IdP or an untrusted IdP since the SP can verify the authenticity of those attributes and can use them to allow users to access sensitive services. In such scenarios, the PPIdP would be truly beneficial.

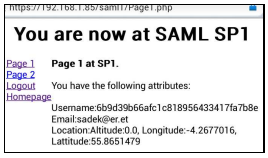


Figure 5. Released attributes in Type 1 Federation.

Figure 6. Two authentication sources at the trusted IdP.

We have deployed another scenario in which the PPIdP can be federated with a trusted IdP to create a Type 2 federation using the similar approach. Here, the fully trusted IdP would act like a Proxy IdP as described in [37] and would delegate the authentication service to the PPIdP which is hidden from the SP. The PPIdP will essentially act as an authentication source for the trusted IdP (Figure 6). The protocol flow will be something like this: the user visits the SP to access one of its resources. The user is forwarded to the WAYF page where the user selects the trusted IdP. A SAML authentication request is sent to the trusted IdP where the user is presented with a selection of authentication sources: The PPIdP (Mobile IdP in Figure 6) or the trusted IdP itself (example-sql in Figure 6). Assuming the user selects the PPIdP, a SAML authentication request is forwarded to the PPIdP and the usual protocol flow takes place. At the end, a SAML assertion with the selected attributes is sent back to the trusted IdP. The trusted IdP retrieves the attributes and creates a SAML assertion with those attributes. To reflect the fact that these attributes have come from the PPIdP, it also inserts a LoA value of 1 for this assertion. The assertion is sent back to the SP. Then the SP will determine which

resources the user can access using the released attributes having a LoA value of 1 (Figure 7). In case, if the user would have chosen the trusted IdP, the assertion would contain a LoA value of 2 and might allow users to access more restricted services (Figure 8).

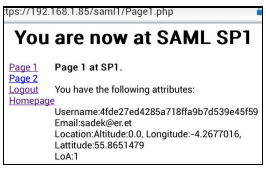


Figure 7. LoA 1 for attributes from PPIdP.

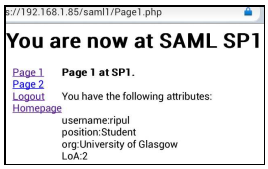


Figure 8. LoA 2 for attributes from trusted IdP.

D. OpenID Protocol Flow

Compared to the SAML, OpenID is a simple protocol. There is no complex trust assumptions to consider as every OpenID provider is considered trusted. There is no need to create any federation using complex metadata exchange procedure in OpenID. A protocol flow for OpenID using our PPIdP is as follows.

When the user visits the SP (Relying Party or RP in OpenID terminology) to access one of their services, the user visits the RP where the user has option to enter her OpenID. The user enters her OpenID (https://localhost:8443/OpenIDTest) and touches the verify button. The IdP is discovered using the mechanism described above (via XHTML form and JavaScript). Once the IdP is discovered, we have opted out the optional key exchange step of the OpenID. Then the SP creates an authentication request which is submitted to the PPIdP using the same approach. The PPIdP traps the request and forwards it to the appropriate servlet. The user is presented with a Login form where the user submits her username/password. Then a response is generated and returned to the RP. The RP then sends a signature verification request to the PPIdP where the signature is verified as per the OpenID specification. Then a verification response is generated and is sent back to the RP. At this point the user is considered as authenticated and the RP takes an authorisation decision to allow/reject the user's request to access the service.

V. Security & Privacy Analysis

A. Security Analysis

Since the PPIdP has been fully developed based on the standardised SAML and OpenID protocols, the security in communication will not be lower than the security of the respective protocol. However, the nature of the deployment introduces several novel security requirements:

Name-Qualifying/Impersonation Issues: The most difficult problem in terms of security for implementing the PPIdP is how to name-qualify different PPIdPs installed in different phones. In general settings of SAML and OpenID, the entity ID of the IdP and the OpenID are name-qualified based on the Domain Naming System (DNS). This ensures that no two IdPs have the same entity IDs meaning that transient identifiers of the users will be different even though two users of two different IdPs might have similar local identifiers in SAML or no two users have same OpenIDs in OpenID. Since PPIdPs installed in different phones will have essentially the same entity ID (https://localhost:8443/saml/idp) or the same OpenID IdP name (https://localhost:8443/OpenIDTest) and different users in different phones might have the same username, malicious users might use this techniques to impersonate users from other phones. The problem can be solved by devising a mechanism to name-qualify each IdP installed in different phones from the viewpoint of each SP. We have developed a Request/Response mechanism to solve the problem. The mechanism is described below for the SAML Type 1 federation:

1) When the user provides the entity ID (https://localhost:8443/saml/idp) of PPIdP and the respective code while creating the Dynamic Federation, the SP generates a random hash and a password. The random hash is to be included in the provided entity ID so that it looks like:https://localhost:8443/saml/idp/random_hash.
   The modified entity ID and the password are also exchanged during the metadata exchange period. The metadata of the PPIdP is based on the modified entity ID which will be used for any subsequent SAML protocol flow. Once the metadata is successfully exchanged and a dynamic federation is created, both the PPIdP and the SP store four pieces of information: the modified entity ID, the password, the entity ID of the SP and the code used for dynamic federation.
2) When the user wishes to access one of the restricted resources, just before the WAYF Page is shown to the user, the SP sends a message to the IdP providing the entity ID of the SP and the code and ask for the PPIdP's entity ID and corresponding password.
3) The PPIdP retrieves the saved entity ID along with the password using the SP's entity ID and the provided code and send them to the SP.
4) The SP matches the supplied entity ID and the password with its database entry. Then the supplied entity ID is used to filter out other PPIdP entity IDs (e.g. entity IDs that belong to the https://localhost:8443/ domain) when the TAL is presented. Note that other entity IDs beloning to other trusted or untrusted IdPs are not filtered out.
5) When the user chooses one of the IdPs, the usual SAML protocol flow resumes.

This mutual authentication between the IdP and SP serves two purposes: i) it helps the SP to name-qualify each PPIdP

and ii) a malicious user will not be able to impersonate other users of different phones.

A SP has, more or less, very little to gain by acting maliciously since a malicious act by the SP will expose its services to unauthorised users. However, a malicious PPIdP has lots to gain if it can impersonate another PPIdP. Using this protocol, a malicious PPIdP would require to know both the modified entity ID and the respective password. Since all communications take place in encrypted channels using HTTPS, the only option for a malicious PPIdP is to guess both the modified entity ID and the password. As both the hash of the entity ID and the password consist of 12 digits of randomly generated alphanumeric characters, they will be difficult to guess. The similar technique has been used to create a SAML Type 2 federation and an OpenID Type 1 federation using the PPIdP.

The whole mechanism is invisible to the user which alleviates them from any associated complexities. Interested users, however, have the option to view the modified entity ID and OpenID provided by each respective SP using the *View EntityID/OpenID* option of the PAS control panel (Figure 3). In cases where the user is accessing the federated SP from other devices than her mobile phone and thus has no access to the PPIdP, our intermediate request/response mechanism will time out and the user cannot view any dynamically added PPIdPs. The presented TAL will contain only the trusted/untrusted list of IdPs. The proposed mechanism has been tested with three different PPIdPs installed in three different Android phones and the result was flawless.

Data Theft: As all user attributes are stored centrally in a database, we had to take great care to ensure that user attributes are stored safely and securely. This is ensured with the help of the SQLCipher library which encrypts every single attributes stored in the database. The database is saved in the external storage of the android phone to allow that both PAS and PPIdP can be access the database. Because of the use of the SQLCipher Library with a secret key (as described in Section IV-A) that only the PAS and PPIdP know, the whole database will be inaccessible by other apps. This undermines any threats involved in cases other apps try to access the database or even if the database gets stolen. The security of the system can be increased furthermore by storing attributes in a hardware/software based TPM. However, the availability of such TPMs for the Android platform is still rare. Therefore it will take sometimes for any wide-spread adoption of this mechanism. Another alternative approach could be the usage of smartcards in which user attributes will be stored in smartcards in an encrypted format.

Intentional/Accidental Corruption of the Database: Since the database acts as the central repository, it is essential to guard against any intentional/accidental corruption of the database. To ensure redundancy, a backup database is always maintained in the internal storage of the PPIdP and PAS which are private to the applications and are inaccessible by other applications, attackers or even to the user. The data in the backup database are stored in ecnrypted format just like the main database. Every single operation (addition/updating of attributes) is synced across both databases. In case the main database gets corrupted, the PAS provides the option to allow the user to restore user attributes from the backup database using the *Restore Data* option of the PAS control panel (Figure 3). Furthermore, users also can create a secondary backup of database to allow them to copy the respective database into some other places (e.g. PC, Laptop, etc.) using the *Backup Data* option of the PAS control panel (Figure 3) and then restore again whenever required.

B. Privacy Analysis

The core privacy requirements for any Identity Management System is to ensure that it supports pseudonymity and empowers uses by transparency and by giving full control over their own data [27]. The support of pseudonymity will enable users to provide pseudonymous identifiers to the SP which will ensure that different SPs cannot collude together to generate a comprehensive profile of the user. The SAML supports pseudonymity with the concept of the transient identifier. The PPIdP adopts the same technique that generates a different transient identifiers for different SPs. This enables each SP to maintain a session with a specific user, however undermines any possibility of linking the same user across different SPs.

The PPIdP uses two techniques to empower users. Firstly, the PAS allows to set a general ARP that will dictate which attributes to release to all SPs. Users also can set a policy for each single SP. Secondly, the user is presented with a form containing the attributes (excluding the attributes that have been filtered out based on the ARP) just before the attributes are about to be released. This allows the user again to fine-tune the number of attributes the user wants to release to that SP. Moreover, the PAS also provides an interface that the user can use to examine the history of which attributes have been released to which entities by using the *Released Attributes* option of the PAS control panel (Figure 3). All these features enable the PPIdP to meet the Explicit Consent, Transparency and Data Control requirements of an IMS.

VI. Discussion

A. Advantages

Our approach offers many key advantages. Here we outline just a few:

- As stated before, this approach empowers users to have full control over the IdP and the attributes stored. Users control which attributes to be released and to which SP.
- The PAS acts as the central repository to store crucial user attributes. Since the attributes are not scattered over many IdPs, their management become easier.

- Users can create Type 1 or 2 federations just in time and whenever required.
- Usually attributes stored in traditional IdPs are static in nature. Mobile phones can be used to generate dynamic attributes such as spatial values that can be used to provide context aware services in a secure way using standard SAML protocol. Such dynamic attributes can also be used to deploy context aware Identity Management.
- Leaking crucial attributes from mobile phones is an important issue. Once an App is granted with a permission to access any attributes or resources, the user cannot control when and for how long that app can access those resources. Storing all attributes in the PAS and binding them under an advanced authorisation framework such as XACML [38] coupled with a SAML IdP (like ours), fine-grained access control mechanisms can be provided.
- By accommodating other IdM protocols, the PPIdP can provide a single interface to interact with all types of SPs.

B. Limitations

Unfortunately, our approach also has some limitations:

- It currently supports SAML Web Browser SSO Profile and HTTP Post binding and so is incompatible with other SAML profiles and bindings.
- Currently our IdP does not support any other advanced OpenID extensions such Provider Authentication Policy Extension (PAPE) [39] and OpenID Attribute Exchange [40].
- The PAS provides a primitive way of setting authorisation. There is no policy based approach to dictate how attributes should be released.

C. Future Work

To harness the full potential of our approach, there are several directions to take from here:

- We plan to add support for other SAML profiles and bindings which would allow our PPIdP to act as a comprehensive SAML IdP.
- We plan to add the functionalities of OpenID PAPE and Attribute Exchange into the OpenID Context.
- Other IdM protocols especially the very popular OAuth (*oauth.net*) could be added to our PPIdP.
- If a user has more than one phones, the current implementation of the PPIdP provides no easier way to share attributes among different devices. It will require the user to create a backup of the database and copy it manually to other phones which could be a complex task. A better approach will be to use cloud services to sync data automatically across devices.
- Adding capabilities to allow the PAS to gather contextual information from external sensors (which are not inside a mobile phone and are installed on the surrounding environments) via network interfaces such as Near Field Communication (NFC), Bluetooth, etc. or using the mobile phone's camera.
- A good research questions would be to find a way for attributes from different sources to be aggregated and stored in the PAS.

VII. Conclusion

In this paper, we have presented a detailed analysis on several aspects of Portable Personal Identity Provider in Mobile phones. We have explained the motivation behind our work, defined several novel concepts, discussed the key challenges to implement those concepts and finally described our implemented proof of concept. As the online services proliferate, it will be increasingly difficult for people to manage their attributes which are currently scattered over many places. A portable personal IdP in mobile phones can be a great tool in aiding users to manage their attributes and will allow users to get back the control over their own attributes which is missing in the current setting. We strongly believe that our approach has great potential and can have a far reaching impact in a world of mobile devices.

References

[1] A. Jøsang, M. Al, and Z. S. Suriadi, "Usability and privacy in identity management architectures," in *ACSW '07: Proceedings of the fifth Australasian symposium on ACSW frontiers'*, 2007, pp. 143–152.

[2] "Shibboleth," http://shibboleth.internet2.edu/.

[3] "OpenID Authentication 2.0 - Final," 5 December, 2007, http://openid.net/specs/openid-authentication-2_0.html.

[4] "Microsoft Windows CardSpace," http://www.microsoft.com/windows/products/winfamily/cardspace/default.mspx.

[5] ITU-T, "Baseline capabilities for enhanced global identity management and interoperability," September 2009, http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=X.1250.

[6] D. W. Chadwick, "Federated Identity Management," in *FOSAD 2008/2009*, ser. LNCS, A. Aldini, G. Barthe, and R. Gorrieri, Eds. Berlin: Springer-Verlag, January 2009, no. 5705, pp. 96–120. [Online]. Available: http://www.cs.kent.ac.uk/pubs/2009/3030

[7] M. Sadek Ferdous, M. Jabed, M. Chowdhury, M. Moniruzzaman, and F. Chowdhury, "Identity federations: A new perspective for Bangladesh," in *Informatics, Electronics Vision (ICIEV), 2012 International Conference on*, may 2012, pp. 219–224.

[8] M. S. Ferdous, A. Jøsang, K. Singh, and R. Borgaonkar, "Security Usability of Petname Systems," in *Identity and Privacy in the Internet Age*, ser. Lecture Notes in Computer Science, A. Jøsang, T. Maseng, and S. Knapskog, Eds. Springer Berlin / Heidelberg, 2009, vol. 5838, pp. 44–59, 10.1007/978-3-642-04766-4_4. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04766-4_4

[9] O. Standard, “Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0,” 15 March, 2005, http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf.

[10] ——, “Web Services Federation Language (WSFederation) Version 1.2,” 22 May, 2009, http://docs.oasis-open.org/wsfed/federation/v1.2/os/ws-federation-1.2-spec-os.pdf.

[11] “OAuth 2.0,” http://oauth.net/2.

[12] “Wikipedia entry on Subscriber Identification Module (SIM),” Accessed on 6 December, 2012, http://en.wikipedia.org/wiki/Subscriber_identity_module.

[13] WP3, Future of Identity in the Information Society, “Study on Mobile Identity Management,” May 2005, http://www.fidis.net/fileadmin/fidis/deliverables/fidis-wp3-del3.3.study_on_mobile_identity_management.pdf.

[14] “Generic Authentication Architecture (GAA),” Accessed on 6 December 2012, http://www.3gpp.org/ftp/Specs/html-info/33220.htm.

[15] D. S. B. Aboba and P. Eronen, “Generic Bootstrapping Architecture (GBA) Framework,” March 2006, http://www.3gpp2.org/public_html/specs/S.S0109-0_v1.0_060331.pdf.

[16] “3GPP specification,” Accessed on 6 December 2012, http://www.3gpp.org/specifications.

[17] B. B. J. R. Marco Casassa Mont and D. Drozdzewski, “On Identity-Aware Devices: Putting Users in Control across Federated Services,” HP Laboratories Bristol, Tech. Rep. HPL-2008-26, March 2008. [Online]. Available: http://www.hpl.hp.com/techreports/2008/HPL-2008-26.pdf

[18] T. Abe, H. Itoh, and K. Takahashi, “Implementing identity provider on mobile phone,” in *Proceedings of the 2007 ACM workshop on Digital identity management*, ser. DIM ’07, 2007, pp. 46–52. [Online]. Available: http://doi.acm.org/10.1145/1314403.1314412

[19] L. A. PROJECT, “Liberty ID-FF Architecture Overview Version: 1.2-errata-v1.0,” Last accessed on 30th June, 2011, http://projectliberty.org/liberty/content/download/318/2366/file/draft-liberty-idff-arch-overview-1.2-errata-v1.0.pdf.

[20] C. Staite, “Portable secure identity management for software engineering,” in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2*, ser. ICSE ’10. New York, NY, USA: ACM, 2010, pp. 325–326. [Online]. Available: http://doi.acm.org/10.1145/1810295.1810375

[21] “Federated Login for Google Account Users,” https://developers.google.com/accounts/docs/OpenID.

[22] T. E. Maliki and J.-m. Seigneur, “User-centric Mobile Identity Management Services,” *Management*, pp. 33–76, 2008. [Online]. Available: http://asg.unige.ch/publications/TR08/ASG2008-3.pdf

[23] “Second Online Security Breach Hits Sony,” 04 May 2011, http://news.sky.com/story/853559/second-online-security-breach-hits-sony.

[24] “LinkedIn passwords leaked by hackers,” 7 June 2012, http://www.bbc.co.uk/news/technology-18338956.

[25] L. Constantin, “Hackers leak 1 million Apple UDIDs,” 7 september 2012, http://www.pcworld.com/article/261869/hackers_leak_1_million_apple_udids_allegedly_stolen_from_fbi_laptop.html.

[26] A. Jøsang, J. Fabre, B. Hay, J. Dalziel, and S. Pope, “Trust requirements in identity management,” in *Proceedings of the 2005 Australasian workshop on Grid computing and e-research - Volume 44*, ser. ACSW Frontiers ’05. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2005, pp. 99–108. [Online]. Available: http://dl.acm.org/citation.cfm?id=1082290.1082305

[27] M. Ferdous and R. Poet, “A comparative analysis of Identity Management Systems,” in *High Performance Computing and Simulation (HPCS), 2012 International Conference on*, july 2012, pp. 454–461.

[28] “SQLite Database,” http://www.sqlite.org/.

[29] “SQLCipher,” http://sqlcipher.net/.

[30] “OpenSAML 2.x,” https://wiki.shibboleth.net/confluence/display/OpenSAML/Home.

[31] “OpenID4Java Library,” http://code.google.com/p/openid4java/.

[32] “SimpleSAMLphp,” http://simplesamlphp.org/.

[33] “PHP OpenID Library by JanRain, Inc.” http://www.janrain.com/openid-enabled.

[34] “Jetty WebServer,” http://jetty.codehaus.org/jetty/.

[35] M. Ferdous and R. Poet, “Dynamic Identity Federation Using Security Assertion Markup Language (SAML),” in *Policies and Research in Identity Management*, ser. IFIP Advances in Information and Communication Technology, S. Fischer-Hübner, E. Leeuw, and C. Mitchell, Eds., vol. 396. Springer Berlin Heidelberg, 2013, p. 131Ãâåœ146. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-37282-7_13

[36] NIST, “Electronic Authentication Guideline: INFORMATION SECURITY,” April 2006, http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf.

[37] D. W. Chadwick, G. L. Inman, K. W. Siu, and M. S. Ferdous, “Leveraging social networks to gain access to organisational resources,” in *Proceedings of the 7th ACM workshop on Digital identity management*, ser. DIM ’11. New York, NY, USA: ACM, 2011, pp. 43–52. [Online]. Available: http://doi.acm.org/10.1145/2046642.2046653

[38] “A Brief Introduction to XACML,” 14 March, 2003, http://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html.

[39] “OpenID Provider Authentication Policy Extension 1.0,” 30 December, 2008, http://openid.net/specs/openid-provider-authentication-policy-\extension-1_0.html.

[40] “OpenID Attribute Exchange 1.0 - Final,” 5 December, 2007, http://openid.net/specs/openid-attribute-exchange-1_0.html.