A secure and dynamic mobile identity wallet authorization architecture based on XMPP messaging infrastructure

Alexandre B. Augusto, Manuel E. Correia
aaugusto@dcc.fc.up.pt, mcc@dcc.fc.up.pt

Center for Research in Advanced Computing Systems (CRACS-INESC LA);
Department of Computer Science, Faculty of Science, University of Porto;
Portugal

ABSTRACT

We are currently witnessing the widespread usage of personal mobile smart devices with serious practical computational power and Internet connectivity. The most popular of these devices being the smart phone, which is in its way of becoming an ubiquitous powerful personal device. At the same time sites like Google and Facebook are deploying an ever increasing set of personal services that are being aggregated and structured over personal user accounts were an ever increasing set of personal private sensitive attributes is being demanded as an excuse for better services interoperability. These user attributes are extremely valuable for these global Internet service companies, as they allow them to produce highly accurate user profiles that they can then monetise very efficiently for marketing purposes. The more accurate a user profile is, the more valuable it becomes, and there are certain kind of personal attributes these companies have just started to harvest that present a major threat to personal security and privacy. These attributes are highly dynamic and are intimately associated to their owners by the means of their personal devices. We have examples like the user GPS position that Facebook uses to make regional publicity or health attributes like the heart beat, body temperature, etc… beginning to be explored by Google at Google Health project. That can be collected and maintained by the user personal mobile devices, sometimes without the user being aware of their disclosure into highly sensitive personal profiles.

In this paper we propose and describe an identity management framework that allows users to asynchronously control and effectively share sensitive dynamic data thus guaranteeing security and privacy in a simple and transparent way. Our approach is realised by a fully secure mobile identity digital wallet, running on mobile devices (Android devices), where users can exercise discretionary control over the access to sensitive dynamic attributes, disclosing their value only to pre-authenticated and authorised users for determined periods of time. For that we rely on an adaptation of the OAuth protocol to authorise and secure the disclosure of personal private user data by the usage of token exchange and new XML Schemas to establish secure authorisation and disclosure of a set of supported dynamic data types that are being maintained by the personal mobile digital wallet. The communication infrastructure is fully implemented over the XMPP instant messaging protocol and is completely compatible with the public XMPP large messaging infrastructures already deployed on the Internet for real time XML document interchange.

Keywords: Mobile Dynamic Personal Identity Attributes, OAuth, openID, XMPP, Mobile Identity Wallet, sensitive personal dynamic data, Security XML Schemas, Android mobile devices.
The massive aggregation of personal identity attributes is currently one of the most important structural and strategic endeavors currently being carried out all over the Internet. Global Internet companies like Microsoft, Google and Facebook are ever more competing over personal user data due to its high strategic commercial value on the market [5], making user digital identity a strategic asset that is going to help to redefine what kind of new innovative services are going to be developed and how they are going to be deployed all over the cloud in an interoperable way. This is well illustrated by the current fierce competition being fought by Google and Facebook about digital identity and their associated authentication, authorization and data exchange protocols like OpenID [15] and OAuth [7].

Currently, identity attributes are normally composed by static values held in the identity management system running in the cloud which can be a bad idea according to Meiko Jensen [13]. What we intend to do with the work described in this paper is to expand the universe of managed static identity attributes with dynamic identity attributes that by their very nature are more intimately associated with their owner and therefore can only reside, not in the cloud, but in mobile personal smart devices that follow their owner everywhere and can therefore keep those values up to date on real time. One good example of a dynamic attribute is the GPS coordinates [10] of a person that owns a mobile device with gps.

What we are proposing is to expand the set of current static attributes being managed and held by Internet identity management systems [18] with a new set of highly dynamic changing attributes. These new identity attributes can be instantiated in classical Identity management systems as symbolic link names that can act as pointers to their real location in the Internet allowing the Relying Party (RP) to locate the digital attribute storage wallets where those dynamic attributes are being maintained and protected.

In this highly dynamic identity infrastructure we are currently developing (Open Federated Environment for the Leveraging of Identity and Authorisation - OFELIA), every time a RP wants to consult the real time value of a certain dynamic attribute it has first to locate the attribute storage wallet where it resides and then ask its owner for permission to access its updated value for a certain period of time, the attribute owner then has the discretionary power to allow or deny that request and provide the RP an OAuth authorisation token, that the RP will present it as proof of previous authorization, every time RP wants to monitor the dynamic attribute during the previously authorised period of time. The attribute owner maintains revocation rights by being able to revoke access at any given moment, thus shifting the balance of power once again to the user, the legitimate owner of the values being monitored and used by cloud services. This is a necessary paradigm shift idea. Highly sensitive dynamic attributes like GPS positioning have high commercial value and therefore access to their updated values should be always put at their owner discretion.

This way privacy is greatly improved by the tools being developed by OFELIA and at the same time users are put into an improved position for bargaining for better services from giant Internet user profiling companies like Google and Facebook that are constantly taking advantages of users profiles.

It is important to realise that dynamic identity attributes constitute a whole new concept of digital identity because their values are constantly being updated due data owner interactions, the RP has to constantly be able to monitor it as requested. This is easily illustrated by the GPS location scenario, where users usually are in constant movement and their locations are constantly being changed, so only with a dynamic identity attributes the RP can obtain the real near current time position of an individual and not the last time the user or application remembered to update it.

Our major motivation for OFELIA is to create a communication infrastructure based on public XMPP infrastructures, network services, applications and API libraries to allow sensitive information [17] such as a GPS position or medical information like a person heartbeat to be exchanged in a secure, reliable and owner controlled way. Keeping this kind of real time sensitive attributes secure and private between the requester and the data owner is a challenge, especially if the framework makes heavy use of a public XMPP messaging infrastructure like the one currently being operated by Google.

We use Android smart mobile phones as our identity digital wallets [20]. Java is native to the Android operating systems so this allows us to have rapidly built a running prototype for user controlled GPS positioning by taking advantage of the numerous libraries, development systems and applications servers which already run on Android. We have used a Java OAuth library [14] to handle the authorisation process to data, a XMPP messaging server [2] to exchange messages between the requester and the data owner and new OFELIA XML Schemas to
validate the authentication, authorisation and identity messages exchanged between RP and the Digital Identity wallet on the mobile device [4].

The rest of the paper is organised as follows. In Section 2, we review the system architecture, describing each node, their functionality and how data flows between the different actors involved. In Section 3 we describe a case scenario of a taxi company using OFELIA framework for tracking passengers, which can be quite useful and helps to illustrate all the messages exchange that has to occur between the consumer and the taxi company to accomplish the task at hand. In Section 4 we described what was accomplished, some preliminary conclusions for the work we have developed thus far for OFELIA, our future plans to the next steps and some development notes about libraries and software that we used to implement our scenario.

2 ARCHITECTURE

In this section we describe in the detail the main components of the OFELIA architecture and discuss the reasons behind some of the options and compromises we had to take to make our vision work into mobile world.

We also take time to describe the flow of data and their important aspects like the protocols and services we have used to build our current secure communication infrastructure. For this stage of development we have decided to limit the OFELIA architecture to the development of two types of services, one for the attributes requester (RP or Requester Service) and other for the data owner (identity digital wallet or Endpoint), assuming that the data and its owner are both engaged in the same mobile device. Figure 1 shows the architecture currently being used for this approach.

![Figure 1: OFELIA architecture](image)

2.1 XMPP: Extensible Messaging and Presence Protocol
XMPP is an open technology for real-time communication that uses the eXtensible Markup Language (XML) as a base format for exchanging information encapsulated into small pieces of XML documents. These XML documents are sent from one entity to other [16], using an appropriate application level transport protocol, normally HTTP, through the means of a rendez-vous XMPP server that relays these messages to the end-points engaged in communication. XMPP servers provide a high standard set of services that can be used by all kind of client applications.

In OFELIA, our messaging infrastructure relies on the following XMPP instant messaging core services [3]:

- One-to-one messaging: This service, allows the exchange of peer-to-peer XML messages. In OFELIA, currently the peers are the Requester Service and the Endpoint Service (mobile attribute digital wallet).

- Authentication: This service ensures that both Request Service and Endpoint Service are authenticated by the XMPP server before start any communication over network, the server acts like a gatekeeper for XMPP network access.

- Presence: This service, grants OFELIA the capacity of checking entity communication availability. This allows the infrastructure to make different decisions based on entity availability. For example the RP can determine whether the user Digital Wallet is on on-line and optional to send an off-channel message (for example an SMS) to ask the device that holds the Digital Wallet to wake up and go on-line to process some urgent asynchronous request.

- Contact Lists: This service allows OFELIA participants to manage a list of trusted entities and thus help a peer to authorise and verify the other peers availability and trustability.

- Peer-to-peer media sessions: This service is currently not being used by our prototype, but this XMPP service, allows a peer to negotiate and manage a media session with another peer, can be really interesting for the future of OFELIA as it enables mobile dynamic attributes to be composed of large and complex audio/video data streams creating new types of usage.

Arguably, in the mobile world, there is some difficulty in directly addressing and communicating with an Internet enabled mobile device. In the mobile world an implicit direct communication with the device is almost impossible due to the shortage of public IPs addresses faced by Internet service providers. In the near future as we can see on [22] IPv6 is supposed to solve this problem, however it is our strong belief that the mobile Telecommunications operators (Mobicomms) will still not allow for this kind of direct communication to mobile phones due to their very inflexible business plans, where the mobile phone is nowadays mostly regarded simply as a consumer device, not a provider of services, in fact most Telecommunications operators restrict even the ports available to initiate communications and the most greedy of them only allows communication over port 80 (HTTP port).

A neutral rendezvous point on the Internet where both RP and Digital wallet can both meet to exchange messages was thus obviously necessary. Towards this end, XMPP proved to be an almost ideal communication infrastructure for OFELIA architecture because of its core services, namely:

i) Almost real time messaging, essential to maintain our dynamic data types exchange between Request Service and Endpoint Service. Dynamic data requests a constantly and almost instant data exchange.
ii) Its ability to operate over HTTP connection by the means of the BOSH (Bidirectional-streams Over Synchronous HTTP) protocol [9], which allow us to bypass the connectivity problems imposed by the overly restricted mobile Internet access from the mobile Telecommunications operators.

iii) Its capacity to store and forward messages in case any of the nodes becomes offline, which is proving to be essential for asynchronous communications. At the beginning of the project asynchronous communications was an important task due the mobile phones sometimes has bad reception of signal resulting on communication problems.

iv) Its scalability to avoid bottleneck problems and the fact that it is a mature fully supported and approved Internet standard, widely deployed and an important part of the communication operations and infrastructure of large Internet operators like for example Google, Facebook, Blizzard and Steam.

2.2 OAuth: Open Authorisation

OAuth is an authentication and authorization protocol originally developed for web applications that provides a standard method for clients to access server resources on behalf of a resource owner. It also provides a process for end-users to authorise third-party access to their server resources without sharing their credentials, using user-agent redirections [7]. The most common analogy to this protocol is the valet key, in other words OAuth works look like a valet key for data access, the one who possess the key have temporary and restricted access to the valet key emitter data.

There are three actors involved in an OAuth transaction: The data owner (User), a third party web application (TP) and the user data storage (UDS). Usually a user wants to provide a TP with an authorisation to access his data that resides on a certain UDS. To achieve this, the TP redirects the user user-agent to the UDS with a formalised request where the user is asked to authorise it, this request includes the data that the TP desires to obtain and for how long time TP wants to access it. After authorisation the UDS returns to the TP a signed authorisation token that allow the TP to access the requested data by presenting then, these tokens can be revoked at anytime by user. The security processes involved in the creation and management of authorisation tokens relies on valid digital signatures and on a shared secret between the OAuth consumer (TP) and the OAuth provider (UDS). An example process fully explained and described can be found on [6].

Currently, in OFELIA both the user and the UDS are at the same place and communicate locally on the same mobile device. Thus, in this case, Oauth communication security is built upon the TP digital X509 certificate and on a session unique key established between the TP and the UDS (Digital Wallet).

This co-location of both the User and the UDS on the same device also have some deep implications in the way the authorisation request and granting process is managed by the means of the OAuth protocol. Since in OFELIA, both user and UDS meet and are located in the same node (the mobile device), when the TP requests access to some identity attribute, an authorisation request appears on the user node showing on whose behalf the authenticated TP is making an access request, what attributes are being requested and for how much time that access must be provided. The user then has to decide whether to grant authorisation, and this can be done in an asynchronous way. Once the authorisation is granted, OAuth will generate and share an access token and a token secret with the TP that must be presented every time it wants to access the authorised user identity attributes. This continues until the OAuth tokens expires or are revoked by the user on the UDS.

2.3 OpenID: Open Identity
OpenID provides a decentralized protocol for user authentication. It is used as an Identity Manager [15] that allows a user to sign into multiple websites with the same account and at the same time control how much of that account user identity attributes can be shared with the OpenID consumer.

Every time a user authenticate into a website (Relying Party (RP)) using OpenID he is redirected to his OpenID provider where he is then asked to login and authorise the identity attribute exchange requested by the website (RP), after that the user is again redirected to the originating RP. In order to standardise and define appropriate semantics for a useful set of user attributes that could be universally recognised by all RPs, the full set of standardised and widely recognised identity attributes for OpenID is substantially reduced. This decreases the usefulness of the protocol and has so far limited its deployment almost exclusively to the authentication domain.

In OFELIA we employ OpenID as an authenticator for the RP (Requester) service. Both the user requester at the Requester (RP) and the user at the UDS need to have a registered OpenID identity that is used in OFELIA to authenticate both identities on the RP. This provides a common account creation and registration process that allows both endpoints to have a common and coherent way of acquiring and verifying identity data.

2.4 Requester web service (RWS)

The requester web service is an integral component of the Rely Party and is composed by a database, an OpenID consumer library and a XMPP HTTP client library. It uses two XML Schemas to authenticate and validate XMPP communications with the UDS and at the same time maintain appropriate data semantics. It also employs two X509 certificates, both emitted by a common trusted PKI, as a way to assure both endpoints (RP and UDS) identity and establish session keys.

When on behalf of a user requester, an OFELIA RP tries to access someone identity attributes, held on an OFELIA digital wallet, the user requester is first asked to login and authenticate himself using an OpenId account. If this is the user requester first login on the RP, this action initiates an auto-enrolment process where the RWS stores, in its database, the requester OpenID address, name, jabber address, user certificate and mobile number, if they exist as OpenID attributes. This account information can then later be enriched with OFELIA OAuth tokens for some identity attributes being held in remote digital wallets, owned by this same requester or somebody else. If the requester has already been enrolled into the RP, he is just authenticated via OpenID and his OpenID identity attributes can then be transparently updated. After login, the RP, on behalf of the current user, can request and try to get identity data from a remote Digital Wallet by the means of the digital wallet endpoint jabber address. If this jabber address is not yet registered onto the RWS, a XMPP message is sent to this address, requesting registration. If the digital wallet jabber endpoint is not reachable, the RWS can nevertheless send an asynchronous request authentication to the digital identity wallet jabber address. This message is held by the XMPP communication infrastructure until the digital wallet comes on-line. If there is a mobile phone number available for the digital wallet, the RP can send a SMS to the smart phone where the digital wallet resides, requesting this endpoint to have his digital wallet to connect into the XMPP infrastructure soon as possible in order to receive the pending OFELIA requests that have been sent by the RP on behalf of the requester, this sms scenario is quite essential since fast battery drain is still one of the smartphone problems [23].

RWS must have a secure storage, because of storing OAuth tokens could be danger if keys fallen in wrong hands. The security details about authentication between the endpoints and the data exchanged are explained at subsection 2.7 and exemplified at section 3 on subsections 3.1 and 3.2.
2.5 Endpoint Web Service (EWS)

The Endpoint service was developed to be deployable on android mobile devices and must take into consideration that the data owner has to personally intervene as a human, during the authorisation process to authorise or decline any request made by a RWS. The EWS is currently composed by a database, both an OAuth consumer and an OAuth provider, a XMPP connector, one X509 certificate to ascertain his identity and two XML Schemas for communication security and OFELIA semantics.

The digital wallet at the EWS must be logged into the XMPP infrastructure with its jabber id and then wait for OFELIA access requests or can be waiting for a sms asking to connect to receive the pending communication. When a request is eventually received, the EWS must validate it against an appropriate XML Schema and process it. In case the requester does not exhibit a valid OAuth token, the digital wallet owner will be asked, by the means of an appropriate GUI, to authorise or deny the access request. This data access requests always contain information about how long RWS wants to access data and a set of specific data that RWS wants to have access. All granted tokens are stored in a database at the EWS along with identity information of whom they have been emitted to, together with an expiration date determined by the wallet owner that can revoke at anytime any token previous granted. Once a token expires their rights are revoked and the only way to renew is through a new data access request from RWS.

Thus, in order for the EWS to receive OFELIA requests from an authenticated RWS, it is mandatory for the RWS to have had a remote user authenticated by OpenID, on behalf of which the OFELIA request is being made and provide the EWS with the relevant identity information needed by the wallet owner to make an informed authorisation decision.

The security details about authentication between the endpoints and the data exchanged are explained at subsection 2.7 and exemplified at section 3 on subsections 3.1 and 3.2.

2.6 XML Schema

XML Schema allows us to define the structure and data types for XML documents, in our project we use OFELIA XML Schemas to help maintain system interoperability between services and be able in the future to decouple endpoint services for different OFELIA implementations [12]. For our framework we employed two different OFELIA XML Schemas, one to handle authentication processes allowing a session key exchange (OfeliaAuth) and the other (OfeliaDataEx) to handle the data exchange and token exchange, it takes place every time a data is requested.

In section 3 we present in more detail an XML exchange flow to better elucidate the documents interchange that can occur in the OFELIA identity infrastructure making it easier to understanding it usage.

2.6.1 OfeliaAuth
As we can see in figure 2, the XML Schema employed for session key establishment and user authentication, consists of a root element called OfeliaAuth composed by three sub-elements: Header, User and Authentication.

The Header element carries information about the state of the authentication and the type of the OFELIA request allowing a clear understand of each communication step. The User element contains personal user information: a Jabber id and an OpenId identity to allow EWS verify user requester identity and the RWS Public key to exchange a session symmetric key in a secure channel. The Authentication element is composed by a challenge string, ciphered with the Endpoint public key to prove its identity and a blank attribute (Session Key) used to return a ciphered session symmetric key.

2.6.2 OfeliaDataEx
As we can see on figure 3 the XML Schema employed for identity data exchange consists of a root element called \textit{OfeliaDataEx}, once again composed by three elements: \textit{Header}, \textit{User} and \textit{Data}.

At the \textit{Header} element we keep information about the state of the data exchange and the type of the \textit{OFELIA} request allowing a clear understand of each communication step like on \textit{OfeliaAuth}. At the \textit{User} element we have the user \textit{Jabber id}, his \textit{OpenId Address} and \textit{OAuth Tokens}, composed by three attributes: \textit{AuthorizationToken}, \textit{TokenSecret} and \textit{ExpireDate}, the usage of these attributes was explained on section 2 at subsection 2.2. The \textit{Data} element is composed by optional elements, describing the nature of the dynamic identity attribute being described. Currently we have a \textit{GPS} element defined with \textit{latitude}, \textit{longitude} and a \textit{timestamp}.

We are currently defining several other elements to describe other dynamic attributes like \textit{heartbeat}, \textit{blood pressure}, etc. That could prove to be useful for remote monitoring web applications. The \textit{Data} element can thus contain highly diverse types of formalised dynamic data types, to cover a highly diverse range of application areas.

In other words, we can provide for all kind of personal dynamic attributes so long as its data type is formalised in the \textit{OfeliaDataEx} XML Schema. It is also mandatory that all dynamic type elements have a valid \textit{timestamp} attribute, not only to be able to maintain an historic value for its values but also to prevent the resending of the same value during different data exchanges.

2.7 Data flow

OFELIA data flow interactions are divided into two main different operational phases:

1- The \textit{OfeliaAuth} handles the authentication process, and it works by exchanging a symmetric session key, using a challenge-response cryptographic scheme ciphered with RWS and EWS public keys, that is then employed to set a secure tunnel between the RWS and the EWS.
The **OfeliaDataEx** phase handles the creation and management of OAuth access tokens and the subsequent identity data consultation.

Assuming both web services have already been logged on into a trusted XMPP messaging infrastructure and that at every step, XML data documents are validate by the appropriate XML Schema, the data flow for authentication and the establishment of a secure communication channel are accomplished by the following three steps:

1. The RWS generates an **OfeliaAuth** XML data document, generates a random challenge string ciphered by the EWS public key from the mandatory previous register on RWS and completes the others attributes leaving only the *session key* in blank. It then sends the XML filled document to EWS by XMPP established connection as can be seen on figure 4a under subsection 3.1.

2. The EWS verifies the requester data, deciphers the challenge and ciphers it again with the RWS public key received on XML. It then generates a session key to set on the attribute *Session Key* and sends it back to the RWS, ciphered with the RWS public key. This is illustrated on figure 4b under subsection 3.1.

3. The RWS deciphers the challenge and if is valid, it stores and sets the received attribute *Key* as a session key to secure the subsequent communication phase of OFELIA. Now RWS and EWS can communicate via a secure channel using the established symmetric session key.

The data flow for OAuth tokens management and data exchange are accomplished in six simple steps, four steps for OAuth token management and the others two steps for identity data consultation.

These six steps are ciphered and deciphered with the established session symmetric key obtained in the previous phase, resulting in a secure channel of communication between the RWS and EWS:

1. The RWS presents an **OfeliaDataEx** document, with a valid *User* element and with an appropriate *Header* to request the identity data and sends it by XMPP to the EWS. This is illustrated by the figure 5a under subsection 3.2.

2. On receiving an **OfeliaDataEx** document, the EWS updates the *header*, sets the *data* types available with an empty value and sends it back to the RWS. This is illustrated by the figure 5b under subsection 3.2.

3. On receiving the partially filled **OfeliaDataEx** document, the RWS updates the *header*, deletes any identity attribute it does not want to consult, sets the attribute *ExpireDate* and sends the updated **OfeliaDataEx** document to the EWS. This is illustrated by the figure 6a under subsection 3.2.

4. On receiving an **OfeliaDataEx** document with the the attribute *ExpireDate* set, the EWS uses the information provided by the document to ask the owner of the identity attributes whether he authorises the consultation of these attributes by the entity whose identity is described in the **OfeliaDataEx** document. If the authorisation is granted the EWS then updates the *header*, generates OAuth tokens for the requested attributes and fill then into the *AuthorizationToken* and *TokenSecret* attributes. It then sends the updated **OfeliaDataEx** document back to RWS. This is illustrated by the figure 6b under subsection 3.2.
5- On receiving an OfeliaDataEx document with OAuth tokens, the RWS updates the header for data request, stores the tokens and their expire date, set the timestamps with the last timestamp received for the requested data and sends the just updated OfeliaDataEx document with OAuth tokens back to the EWS. This is illustrated by the figure 7a under subsection 3.2.

6- On receiving an OfeliaDataEx document with OAuth tokens and a timestamp, the EWS updates the header, verifies the validity of the presented OAuth tokens, and fills the data element of the OfeliaDataEx with an array of dynamic type elements with the historic set of values the dynamic type has assumed on the Digital Wallet since the timestamp just received on the OfeliaDataEx document. It then sends the document back to RWS. This is illustrated by the figure 7b under subsection 3.2.

3 USAGE CASE SCENARIO

In this section we are going to describe a concrete case scenario to help a better understanding and illustrate how the OFELIA XML Schemas can be used within the XMPP infrastructure for a RP to get access authorisation and then subsequently present authorisation tokens to monitor dynamic attributes from a remote digital wallet held on a mobile device. For illustration purposes we are going to explore a real case scenario where a web application is authorised to monitor a user by the means of the GPS device he has on its mobile phone.

Let us assume that a taxi company, let us name it "We know where you are", decides to use our system to implement an innovative service on the Internet for its customers.

The taxi company releases a mobile application with our endpoint OFELIA web service for smart phones to track clients GPS data. Every time a passenger needs a taxi he utilizes the application to request the service, after that the company cloud application requests the passenger for authorisation, using the OFELIA infrastructure to have for the next quarter of an hour temporary access to the customer digital wallet to track the customer geographic location thus allowing the nearest taxi driver to find him. This solves problems like unknown roads and moving customers. Since the tracking authorisation is temporary the customer privacy is protected because that information is only disclosed when the customer is in need of service and everything is kept under his own control in other words the passenger can revoke the authorization to access their location at any time.

For a better comprehension of the data flow we are going to present a simplified XML OFELIA flow of documents for this scenario, the taxi company Internet cloud application acts as an OFELIA RP and the mobile application on the customer phone acts as an OFELIA digital wallet.

3.1 XML data flow for Authentication

A customer decides to request a taxi using the mobile application. The taxi company then acts as a requester web service and sends a XML document request with a challenge, the mobile processes the XML document request and replies with a symmetric session key that will be used to secure all future communication, as illustrated on figure 4.
3.2 XML data flow of data exchange:

After the establishment of an OFELIA session, the taxi company application sends a request to determine what types of dynamic identity attributes the customer holds in his identity wallet. The mobile phone processes the request and sends the XML document back with a response, where the Data entity is properly filled with the relevant information. This process is illustrated in Figure 5.

The taxi company requester now deletes from the XML response every data type supported by the remote digital wallet, except for the GPS dynamic identity attribute. It then sends the updated document back to the mobile phone, requesting access OAuth tokens with an appropriate small expiration date. On reception the customer is asked by the mobile phone application to accept the request and grant access to the GPS. The mobile phone then sends the request...
response back to the company application with valid OAuth tokens. Figure 6 illustrates this process.

![XML Data Ex (a) Request and (b) Response](image)

**Figure 6: OFELIA XML data from tokens request**

With the tokens now stored on the company application, the taxi company now sends an OFELIA request each 5 seconds to monitor the customer GPS location data. If the OAuth authorisation tokens are correct and correspond to previously established sessions, the customer mobile replies with an XML response with the GPS entity data correctly filled with updated data. This information is then relayed to the nearest taxi driver that can thus temporarily track the customer and find him in the shortest possible time. This can be observed on Figure 7.

![XML Data Ex (a) Request and (b) Response](image)

**Figure 7: OFELIA XML data from Data Exchange**

### 3.3 Reverse taxi scenario
Let us assume that the taxi company "We know where you are" made a huge success with this system and now they are studying the possibility of a reverse scenario, in other words both taxi and client will know their current position. As we already saw in our case scenario this case is completely possible.

The taxi company will have to update both the company cloud application and the mobile application; these applications will have to do EWS and RWS service at same time at both nodes to allow mutual tracking.

Every time a passenger needs a taxi he starts the taxis company mobile application to request the service, at the same time the mobile application demands access to the taxi GPS data, now the company cloud application requests the passenger for authorisation to track his position for the next quarter of an hour or until both nodes find each other in a single point, after passenger grants permission the company cloud application have to assign the nearest taxi and authorise GPS data track of the assigned taxi to the passenger. Now both the passenger and the taxis driver can locate each other and if passenger wishes he can walk to into taxis direction to speed up the encounter.

4 CONCLUSIONS

In this paper, we have described how the OFELIA XMPP communication infrastructure implements a user empowering way of disclosing sensitive dynamic attributes held on mobile devices. The whole process relies as much as possible on standard well established services and protocols like OAuth, XMPP and openID, thus allowing for a simpler implementation and rapid service deployment. We are currently developing a working prototype composed of a Rely Party and a digital wallet held on a Android mobile devices that uses the Google XMPP messaging service to exchange documents in a secure and private way. Our prototype allow the Rely Party to track the mobile phone user real time GPS position after a previous authorisation by mobile device user and plots its location using the Google map service for the periods of time authorised by the user.

We introduced a new concept of data type that we named dynamic identity attribute (DIA), this new type of data opened a whole new range of opportunities and possibilities due the ability to allow data be processed as requested (PaR), in other words every time a rely party requests the digital attribute storage wallet a DIA this data value is processed in real time, for static attributes DIA does not present great advantages (unless data is sensitive) since static attributes doesn’t need constantly update but still has the advantage of the user being your own data storage and share your information only with third parties of his trust, however this scenario have a considerable change and become essential when we use DIA with volatile data, because of its high constant value change, PaR becomes mandatory to keep a coherent timeline of data requested.

This digital identity infrastructure comes at a time where there is a real need for the users to gain back some control of their own privacy and only disclose their most sensitive identity attributes when they need a service from the Internet that really requires access to this data values to work, and this for only a very limited period of time, kept under strict control by the user data owner.

With our OFELIA prototype we have also proved that XMPP, with its very rich set of working extension services, constitutes an excellent choice for network communication, allowing us to quickly set up a system ready to manage and implement complex data exchange processes in almost real time. With XMPP problems like connection restrictions, offline messages or security are readily solved in a standard way, as has been previously described at section 2 on subsection 2.1.
4.1 Future work

In the near future OFELIA will have planned a several tasks to implement, namely:

**EWS on mobile phone:** OFELIA is still running on desktop computers, with the digital wallet set to run on an Android emulator. The next step will be to deploy the EWS directly into a real android mobile phone with a GPS. We have already did feasibility studies and experiments about running web services on Android devices and found the web container called i-jetty [1] to be good choice to run the EWS on a mobile phone.

**RWS and EWS as one:** At this moment our identity digital wallet only generate and manage their own access tokens working only as a personal secure data storage but in a near future we plan to expand the wallet as a data requester too, allowing users to request data too. This expansion is quite interesting and important to OFELIA success; users will have the possibility to request other users for their data letting users share information like GPS position in real time, problems like lost friends in roads will never happen anymore.

**More Dynamic Identity attributes XML schemas:** We want to expand OFELIA to other application areas. The eHealth sector [11] is one area where there is a real need to remote monitor patients in a secure privacy oriented way. This is currently done with highly expensive equipment with no provision at all for the patient privacy. With OFELIA we want to be able to the same thing in a better way, by taking care of the patients privacy and the same time provide remote monitoring services with much less expensive equipment, mainly composed of Android phones and Bluetooth compatible body sensors capable of monitoring the patient heart beat, body temperature, blood pressure, etc…

**Define others authentication methods:** After some research we are studying new possibilities of authentication like the mobile secure card(MSC), this microSD card brings the usual flash memory plus a smart card chip giving us a strong authentication due the advantages of a smartcard explained at [19], we are studying too a possibility of the usage of Portuguese citizen card via a Bluetooth to authenticate.

**RWS as an API:** To allow for good extensibility, usability and portability of the system it is essential in the future to program the RWS with the help of an easy to use decoupled API for an easy integration of monitoring services into many different web applications without the programmer having to know about the intricate details of OFELIA XML document interchange and OAuth tokens.

**Symbolic links to sensitive data:** As we already said we want to establish a formal way for this symbolic links aiming standardization over network. After a service get a symbolic link it will be possible to that service know what type of data it is requesting and what is the next hop it needs to ask about the desired data.

4.2 Development notes

As already mentioned to implement OFELIA architecture we relied in some libraries and software. In this sub section we exposed the libraries, their versions and if their respective links:
**OpenID consumer library:**

Openid4java
Version: 0.9.5.593
Download link: http://openid4java.googlecode.com/files/openid4java-full-0.9.5.593.tar.gz

**OAuth provider and consumer libraries:**

Oauth - API needz authorized?
Revision: 8278
Download link: http://code.google.com/p/google-web-toolkit/source/browse/tools/lib/oauth/?r=8278

**XMPP BOSH client connector:**

Ignite realtime SMACK API
Revision: 12894
Svn link: http://svn.igniterealtime.org/svn/repos/smack/branches/bosh/

**XMPP test server:**

Ignite realtime Openfire
Version: 3.6.4
Linux download link: http://www.igniterealtime.org/downloads/download-landing.jsp?file=openfire/openfire_3_6_4.tar.gz

Acknowledgments

This work is being financed by the FCT funded project OFELIA (PTDC/EIA-EIA/104328/2008), and is being conducted with the institutional support provided by DCC/FCUP and the facilities and research environment gracefully provided by the CRACS (Center for Research in Advanced Computing Systems) research unit, an INESC LA associate of the Faculty of Science, University of Porto.

**REFERENCES**