D-FOAF - SECURITY ASPECTS IN DISTRIBUTED USER MANAGEMENT SYSTEM

Slawomir Grzonkowski*, Adam Gzella*, Henryk Krawczyk¥, Sebastian Ryszard Kruk*, Francisco J. Martin-Recuerda Moyano¤, Tomasz Woroniecki*

* Digital Enterprise Research Insitute, NUI Galway, Ireland
e-mail: firstname.lastname@deri.org

¥ Gdansk University of Technology, ul. Narutowicza 11/12, 80-952 Gdansk, Poland,
e-mail: hkrawk@eti.pg.gda.pl

¤ Digital Enterprise Research Insitute, Leopold-Franzens Universität Innsbruck,, Austria
e-mail: francisco.martin-recuerda@deri.org

Summary

The contemporary Internet offers various services ranging from electronic newspapers to on-line social networks. To authorize themselves, users have to register to on-line services. However, most of the authentication and user management systems are incompatible with each other. Therefore the registration process must be repeated each time from the beginning, requiring multiple login-password-site triples with adequate security constraints. Very often, user management systems do not allow user to view or manipulate their profile information, and so users cannot determine the actual information gathered about them after registration. To overcome the problem with multiple registrations and sign-ons, a number of solutions like Microsoft Passport have been proposed.
In this article we elaborate on potential security risks concerning single registration, single sign-on and access to profile data. We present how required security levels in a user management system can be provided without losing the accessibility of the service. We define how the potential user can benefit from this user management system based on open standards and open architectures. Finally, we present the D-FOAF, a distributed user management system based on FOAF metadata and a P2P architecture that implements presented solutions for secure distributed user management system.

1. Introduction

Business to customer communication has never been easier before. To set up a successfully enterprise one does not need to rent expensive trading space, a web site seems to be enough. The virtual world that has been set up just beyond the screen of our computer covers many aspects of human life activities. From interpersonal communication, through on-line shopping and flights booking, to e-banking we are equipped with a hand full of services ready to be used.

The only problem is that in order to use them we need to register. Very often users are soon getting bored after submitting the same information for so many times during the registration process. To ease the burden of repetitious registrations a number of solutions has been suggested. Among them such a famous ones as Microsoft Passport[1].

Along with the Internet standardization efforts, W3C has proposed FOAF[2] vocabulary for RDF[3] that covers the basic set of properties used across different user registration forms. RDF information can be easily distributed, shared and reused. FOAF

1 Microsoft Passport: http://www.passport.net/

2 FOAF: http://www.foaf-project.org/

3 RDF: http://www.w3.org/RDF/

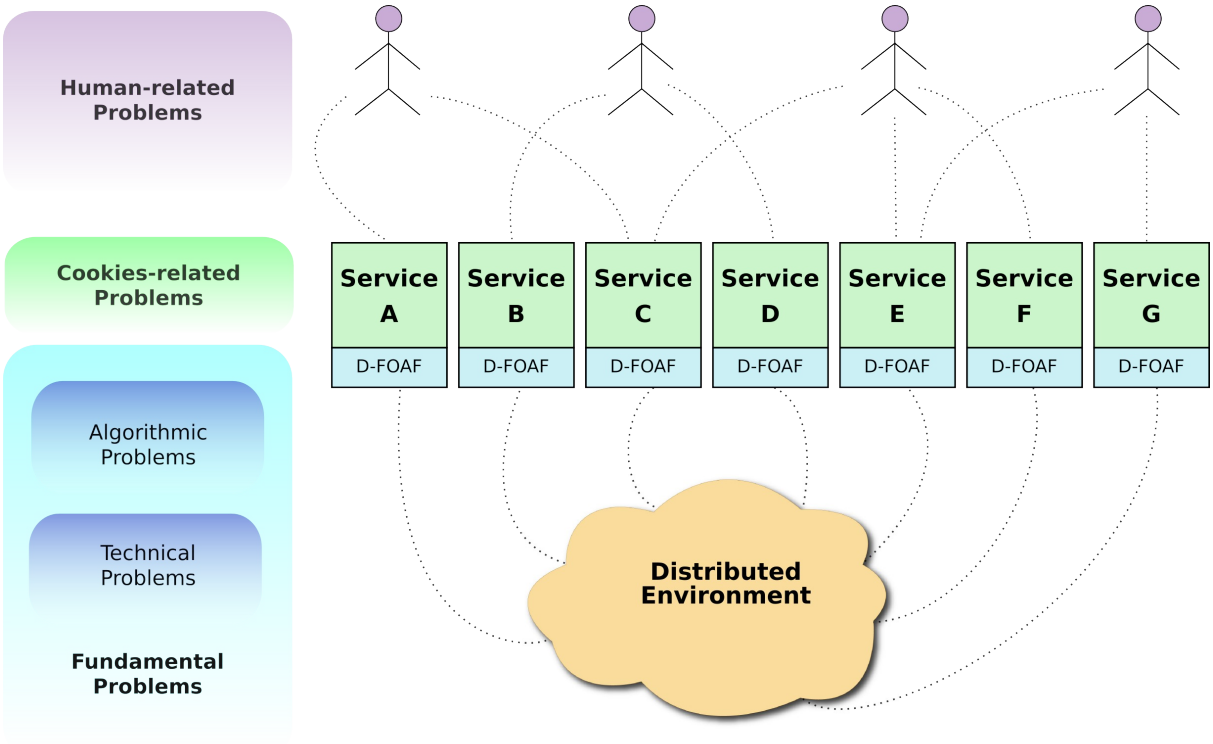

Figure 1: Security problems in distributed user management system

starts to be gradually recognized as an Internet description format on the user profile information. So far, there are not many, popular user-end applications yet. The FOAFRealm[4] project[4], the predecessor of D-FOAF[5], adopts FOAF vocabulary for user profiles management. The FOAF file can be used during registration process, taking away all the hassle with filling in all registration fields.

1.1. Distributed User Management

Many websites use naive approach to user management. The 'management' is often limited to storing name and password, offering only static and predefined access control. In most cases the predefined privilege groups are flat, without any relations to each other.

Very often it is not possible to define new roles, groups and relations in the run-time by users themselves. The systems like Orkut[6] introduce the notion of determining access rights to parts of user profiles by users themselves. But the list of possible options is limited to: myself, my friends, friends of my friends and everyone.

Most user profile management systems supports only one service at a time. It is very hard or almost impossible to use the same profile across different services. Systems like Microsoft Passport aim to support re-usability of the profile information across different services. Centralized topology, proprietary solution and very frequent bug reports are some of the reasons why Passport has not been yet widely accepted by the Internet community.

Sxip[2] overcomes drawbacks in centralized architecture of Microsoft Passport. The users gain more control over their profile information stored on one of home servers. Though it is a step forward with respect to Passport, Sxip is centralized from the perspective of home server. This keeps back all services that requires additional and sensitive information to be treated more securely.

The ideal solution would be a distributed architecture of underlying user profile management systems. The P2P topology is an ideal candidate[3]. There is not central point of failure and profile information can be easily gathered across the network without hassle of finding the profile information providers first.

On the other hand, managing distributed user profiles requires solving some security problems. One of them is the process of locating the home server of the user that is being authorized. The

4 FOAFRealm project: http://www.foafrealm.org/

5 D-FOAF: http://d-foaf.foafrealm.org/

6 Orkut Social Network: http://www.orkut.com/

presented credentials must be checked with the home server only. It will keep the required security level even in case of appearing of a malicious host inside of the network.

In addition to security in single-sign-on environment delivered by distributed user management system users and services need to have both control over the profile information and fine-granularity of security constraints. Users want to restrict access to some of their sensitive, private information only to some classes of services. A flexible solution will give a chance to choose policies like 'always ask' or 'insurance number allowed only for government institutions'. A service provider such as a bank will need much more assurance about clients identity than a news group server. Some of those solutions has been proposed in PeerTrust[12].

1.2. Outline of the Article

The article is organized as following. Section 2 presents on possible security threats. Followed by section 3 analysing possible solutions security aspects in distributed user management systems. In section 4 a distributed user management systems, D-FOAF, is being presented. Section 5 presents related work. Finally, section 6 concludes on the usability and future of distributed user management systems.

2. Potential Security Risks in Distributed User Management Systems

Providing security in distributed user management systems is an essential task. Unfortunately, many users treat it as unimportant problem and carelessly believe that every program does not have any malicious intentions. The aim of this chapter is to consider the most important threats both fundamentals and widely known for many software and specific for ideas related to D-FOAF (see Fig. 1).

2.1. Fundamental Problems

Algorithmic. These class of risks can often solved in algorithmic methods. Although a lot of services do not take the problem seriously, especially non-commercial products.

Authentication - The system needs a way to uniquely identify each peer. Granting access rights should be based on information the system has on service requester. This problem is very often solved by register forms and secure communication between client and server.

Encryption Cracking - Currently, the biggest problem is brute force attack. Although, protocols use strong algorithms, computing power increases very rapidly. Moreover, it is easier and easier to build a cluster of supercomputers and take advantage of the technologies like PVM or MPI to break logins and passwords faster.

Man in the middle - If an attacker is able to observe and intercept messages exchanged between two potentially victims, he can easily read, modify and insert content at will. Messages between two parties without either part knowing that the link between them has been compromised. This method works even if the data is encrypted. Nowadays, several solutions have been proposed although they make algorithms slightly more complicated.

Replay – Very often data in distributed environment is transferred through other. It causes well-known problem. A malicious peer is able to read the information and use in another services, even if the data was encrypted.

Technical. Problems described below are very often related to TCP/IP. The protocol was developed in order to survive external threats like nuclear attack. Unfortunately, many of problems are caused by internal intruder.

Denial of Service - Systems based on single sign on protocol are especially susceptible to this kind of attack. Usefulness of these systems increases in proportionally to the number of users who are subscribed. These services might be simply flood by an attacker with a random profile registration or logins. The common response to such problem is to distribute and to replicate the service. Although, it seems to be right the solution, it has also significant drawbacks e.g. many copies of the secure information.

Replications - Replications seems to be a clever remedy for the DoS attacks but replicating the service require multiple copies of the secret information like keys and profiles data. So it gives an additional opportunity to compromise the servers.

ARP, TCP Spoofing - There are many possible types of this attack. Attacker can change IP address in the datagram. Moreover many manufacturers of ethernet cards provide software that allows to easily change MAC address and in result pretend another machine. In most cases administrators can hardly detect intruder.

DNS Spoofing - Intruder who controls client DNS can simply perform aliasing of one address to another which is controlled by an attacker. It creates an opportunity to provide user with fake web site that mock-up the original login form in order to steal secret information.

Sniffing - Sending logins and passwords as plain text is a frequent mistake. Tools like *ethereal*, one of the most popular sniffer, allow users to capture and analyse packets passing through their computers. Unfortunately, there is no need to be an expert to capture or understand stolen data. Opportunity depends on available tools and used protocols.

Centralized Data - Many single sign on systems uses one centralized data store, which is obviously a bad idea. A successful attack would be particularly disastrous that will force us to change each password.

2. 2. Cookies Problems

Nowadays, many problems are caused by cookies. The idea of the cookie is to free the user from retyping many times the same data. Sometimes it brings about many unexpected problems. Since new protocols and applications are still being developed, there is opportunity to find new bugs exists.

Persistent Cookies - Services like single sign on leaves authenticators in the cookies on the client machine. As a result, users do not have to retype their passwords during the next visit. If the client have proper cookies, no further proof is required. Anyone who gets the cookie has unlimited access to the users private data. Moreover, if the user forget to log out from public machine, he can not to do it remotely and he have to wait until the cookies expire

Cookie Monster Bug - Cookie Monster bug is an example of a dangerous bug which affects some versions of Netscape Navigator (4.51 fixed it). The bug allowed to malicious server applications to set a cookie on the browser at higher domain which should be forbidden. As a result, the cookie was sent to any server in the higher domain which browser connect to in the future. Cookie Monster bug caused a specific Denial of Service attack.

2. 3. Human-related Problems

These problems are based on the fact that many people are not skilled in computer science aspects. Furthermore, they almost always believe in good intentions of programmers. Unfortunately, as long as the Internet exist, malicious users will be going to get access to private a data and a secured information. Although new protocols and more reliable application are developed, new security problems arises.

Signing out from multiple services at the same time - When inexperienced user decide to log out from a website, which use single sign on protocol, he/she is not aware that persistent cookies are still on the hard disk, especially if he/she sees log out message on the screen. Website client should be informed that leaving one site does not mean to log out on single-sign-on server.

Weak Passwords and User Names - It is a very well-known problem that users tend to pick poor user names and passwords that are guessable or too short. Furthermore, they keep using them on different servers. It is an opportunity for both malicious administrators and hackers to gain control over another the server.

Distributing Key Methods - In order to provide fully reliable service the way keys should be transferred via e-mail or phone. Every information in the network can be sniffed and stolen somehow. In reality, only some services like Internet auctions use this method of distributing keys.

Java, JavaScript - These scripts and applications often try to imitate dialogue windows. Inattentive user will fill in the form with login and password. Afterwards applet can easily send the private data to the intruder.

Incorrect Spelling of a Search Services - Inattentive users are especially susceptible to this attack. Attacker set up own website in order to imitate commercial service. Although, there is no visual difference between these too web pages, URLs are slightly different. Unfortunately, it is very easy to make mistake and allow our login credentials to be discovered. Incorrect spelling addresses are usually distributed by spamers via emails (*fishing*).

2. 4. Other Problems

The following risks are related to all kind of software. They can turn casual applications into extremely dangerous enemies.

Software Bugs - If the distributed environment software contains a bug, it can expose the network to a risk. Conflicting applications might cause the system to crash or decrease its overall perform-

ance. As mentioned above, Cookie Monster bug caused DoS attack.

Trojans, Viruses or Sabotage – Distributed systems very often route communication through the firewalls. As a result, if the intruder is able to control the firewall, he can launch DoS attack, gain control over the network resources or simply steal private data.

3. Security Solutions for Distributed User management Systems

To summarize previous paragraphs, we could divide the security problems into some groups:

Technical problems - In this group we can find the majority of the problems described beneath. These are the threats refer direct with Internet and its protocols.

Logical (functionality) problems - here we would see all the problems related to the main functionality of the system. We need to answer some very important questions e.g. How to authenticate user? How to pass user attributes in distributed FOAF-nodes? How to store the required information on the client side?.

Client side problems - are related to uncertainty and unpredictability of human behaviours. In this group we could also identify problems with cookies stored in the user's browser.

All of this groups need a certain amount of actions and solutions.

3. 1. Solutions to Technical Problems

As it can be seen above, there is a lot of the very detailed threats connected with Internet. For most of them there are well known solutions. A lot of problems can be solve by using secure (SSL) connections in communication between servers and between the server and the client. This is widely used and very popular solution, which eliminates a lot of potential network risks. Below we present some example solutions for the most common technical problems:

Encryption cracking - the best and probably the only solution here is to use relatively (over 1024 bits) long keys.

Man in the middle – a number of cryptographic solutions has been identified to handle that problem.

DNS Spoofing - the easiest way to defend against DNS Spoofing is to base the communication on the IP address rather then DNS names.

ARP,TCP Spoofing - The best protection against this type of threats is to use standard cryptographic methods like SSL with key verification. In case of distributed environments cooperation with well established certificate authority (CA), like VeriSign[7] could be an option.

Replay - Main question here is how to send the user credentials through the network in the way that it could not be used by other peers. This problem is related to the other issues (i.e. the cookies) so it will be discussed in the following parts of paper.

Lot of the threats are mineralised naturally thanks to D-FOAF construction. There is more than one server with users profiles, so attack to the one server would not destabilize the network (DoS attacks, centralized data, etc.). The main challenge is to adopt mentioned, well known solutions to our particular project.

Very important issue is also the connection between FOAF servers. It is based on HyperCuP technology, which provides some security solutions and our project depends on this technology.

3. 2. Solutions to Algorithmic Problems

This are probably the most important security problems in D-FOAF. We need to provide the secure single-sign-on process which is quite complex, especially in distributed service network (so without central authorisation service). Since even the strongest means of security provided by software applications fails on human based attracts[8]. We need to find the way to transfer user credentials through the network, properly authorize the user and allow him to log-in to the other service without giving the login and password.

There are three possible solutions for user authorization process:

- Distinguish some servers as a trust, authorizing servers. Such a solution introduces a fundamental problem of violating assumptions imposed on ideal distributed environment.
- Provide the client side applet, which would gather user profile information. It could properly encrypt the login and password, and then send it to the network without any

7 http://www.verisign.com/

risk. The main thing here is the problem with interoperability. There is a need to provide different solutions for Java and .NET platform. There could be some issues with various operating systems.

- Divide the authorization process into two steps - in first step user gives his/her login part of credentials (e-mail address or mbox_sha1). Then the system finds the home server (the one with user's profile information). As soon as user login will be passed, the browser will redirect the user to his server. Then user will be prompt to give his password. After successful authorization a cookie is stored in user's browser. The information about the user is then send back to the invoking server.

Next thing is the situation when system need to authorize the user which was logged by the other service. There need to be some information kept in a cookie. Decision should be made on what to store in a cookie and how crucial this information is. It should allow the user to log in to the system, and should be useless for other pairs (which can read this cookie).

We need to keep user's home server address and encrypted data (e-mail with password). Additionally there could also be client's IP address remembered on the server. (and also encrypted into cookie with login and password). That would protect the user from attempts to steal the information and acting like this particular user.

When the service identifies that the user has D-FOAF login information it should read the address of the home server and redirect user to this server. The home server should proceed with authorization process just like for the first time using the information taken from cookie.

In the next section we describe more solutions related to cookies problems.

To achieve better security, the key which was used to code the data, should be different for every user. Without that solution breaking the server key would provide a great danger to the whole network.

3. 3. Solutions to Client-side Problems

Maturity level. One of the idea to solve so *human* problem is to add to the user profile some information about his, so-called 'computer maturity'. In one way it could be understood as a something similar to the debug level. E.g. if a user wants to know what is happening with his personal details, on what of authorization stage he is, he could be informed about every event in the application. On the other hand the additional information could be used to set the proper expire time to the cookie (that will keep user data on the computer). So if user often log-in from different computers, (for example Internet cafés) he could set his maturity to low level. That will set a very short cookie expiration time and would decrease the danger of taking over his account by somebody else.

Counting sessions. The abbility to remote log off from the application would be a very useful functionality. After some considerations we recognized that probably the best, would be one, very simple solution. As we know the information about that user is log-in, would be kept in cookie on the client side. Why not to keep there also some counter of how many times users logged-in into FOAFRealm system. The same number would be kept on the servers side - with users profile. Each time a user logs in a counter value in his/her profile is incremented. Assuming that user did not log-off from one computer and he want to do so on the other, he/she could simply log-in one more time (of course on the interface level it could be properly masked), the counter value would change in his profile on the server, so other calls using different counter value would be refused. This issue has very simple, yet efficient solution which not only allows user to log-off but also assured that only one - actual session is valid, and there is no need to worry about any of the old sessions.

Of course the counter must be suitably encrypted to avoid any manipulations. It could be also indirect, for example server could randomly generate some next states of the counter. It should work like this since we have assumed that all the sessions with counter value that does not match the actual one are rejected.

Webmasters' responsibility. Unfortunately the mentioned above are attempts reduce the danger for the user only. Crucial here is the user interface. It should be very clear, and his responsibility is to remind user to log-off before leaving the computer. Of course this facts is not in the D-FOAF interest. Despite this we should remember about this threats and we need to create some guidelines for the webmasters creating systems with D-FOAF functional.

Giving opinions. During our research we were considered giving opinions to differentiate the se-

curity level parameter between services. There are many possible solutions. The most important question is who should be able to judge the services and users. Two main components of distributed user management environment.

The users represents casual people who use the Internet. We do not have too much information about them, usually they create their profiles by filling in forms on the web-pages.

The services are persistent part of the network. To create and popularize a service much more steps are required. In our concept the part of the network which get only one negative should probably be removed from the network. So instead of giving opinions peers could ban other peers. However it could be useful to keep amount of opinions (positive or negative), so they can be used to create user rank.

Even though the problem who should have a right to judge other network users still exists. Below we present some possible solutions of this problem.

Services to services - This proposal describes giving opinions among services. Again, the most important decision is which services should be able to judge. Unfortunately, we can hardly do it automatically. We may decide to use page rank or the information on how many times the service has been used. The criterion seems to be reasonably. However, they causes many difficulties. In most cases banks are less popular and less visited than illegal video or MP3 sites. The example has been presented that we can not do it automatically in trusted networks. We must choose reliable services at the beginning of creating network and give them ability to grant and revoke privileges for the others.

Finally, if we managed to select the judges, service owner would have to specify a threshold. As an example, the opinion might be positive (+1) or negative (-1). If the sum were below the threshold, user or service couldn't use the service.

Services to all - It would be much more helpful if services were able to store opinions about users. This information would be shared among services. Some of them like banks would be able to send broadcast query to the network and gather all opinions about specified user, especially before perform operations like create account or transfer money.

Although the idea seems to be interesting, the problem is how a service can give opinion about user? This is great area for new ideas. As an example, if system founds two users with equal passwords and IP addresses of the creator, it might assume that this is a multi-account which is not allowed in many services, as a result system will give negative mark. If user attempted to fill in a form with JavaScript code or html tags, system would give him negative opinion. Positive marks could be given for long time work without breaking internal rules.

To sum up, giving opinions seems to be an interesting idea and it is possible to use it in the future. However, many related issues require development and additional research.

4. D-FOAF – Distributed User Management System based on FOAF

The D-FOAF project aims to deliver a distributed user profile management system that covers issues presented in this article. D-FOAF is based on highly scalable distributed HyperCuP P2P infrastructure[8].

In comparison to existing solution like Microsoft Passport, D-FOAF is flexible and user oriented. MS Passport does not allow to define groups of access rights or connections between them. Its role in social networking services is therefore limited. D-FOAF is a step forward towards security without central authority. It does not solve all issues with distributed authentication, but proposes a completely different direction than Passport.

FOAFRealm, the predecessor of D-FOAF, addresses the need of fine-grained access control. It may be used to enable end users to define their own policies, and eventually create collaborating communities. Additionally, FOAFRealm is based on the FOAF vocabulary and RDF storage. This provides users with more control over their profile information, even though it is being distributed between various services they use.

4. 1. The Architecture of D-FOAF

D-FOAF binds many different services together into P2P network. The underlying topology of P2P communication between D-FOAF instances is based on HyperCuP infrastructure[10]. This en-

8 Lightweight HyperCuP Implementation Project: http://www.hypercup.org/

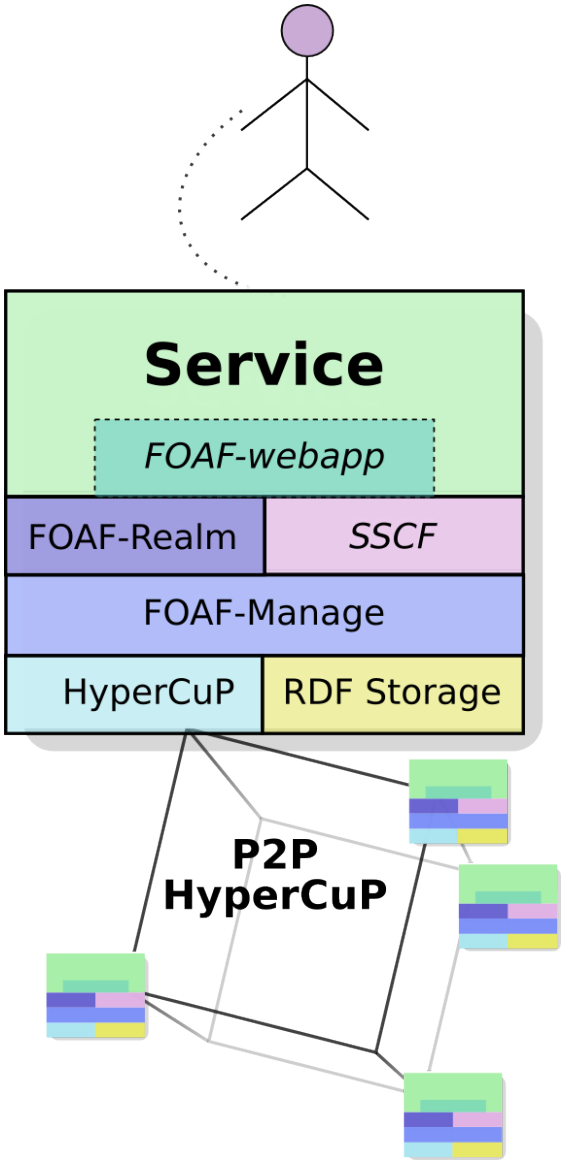

Figure 2: The architecture of D-FOAF

sures high level of scalability and fault tolerance.

The single D-FOAF instance service stack architecture (see Fig. 2) consists of low-level, lightweight HyperCuP communication and RDF Storage (Sesame at the moment) at the lower tier. Information from both sources is managed (FOAF-Manage) and presented to higher services:

- *FOAF-Realm* – An authentication component that performs actions required by the actual service during the user authorization procedure
- *SSCF* – Semantic Social Collaborative Filtering implementation, that enables registered users to disseminate and acquire knowledge across the social network.
- *FOAF-webapp* – delivers a set of DHTML, JSP, Java code, ready to use in the user interface of the service

During the registration process user may use his/her FOAF file. Once registered to one of the services (also called home service) associated in D-FOAF network, the user can access the protected resources the same way as he/she would be registered to the each of them independently. The main difference to similar approach in Sxip is that user does not have to supply any other information to non-home service. The home service locate automatically across the P2P network. Additionally user has full control over the profile information gathered across different services.

4. 2. FOAF Graph as a ACL Back-end

One of the key features in D-FOAF/FOAFRealm is fine-granularity of access control lists based on FOAF/RDF graphs. There is no predefined list of groups, realms of users.

Figure 3: Example of ACL entry in FOAFRealm

Each group is based on the distance and trust level from person A to person B. Where person A (a root person) is a arbitrary selected person in FOAF graph, might be a virtual one, representing a group. And a person B is trying to access the protected resource. The minimal distance and maximal trust level from person A and person B is calculated over the FOAF digraph with Dijkstra algorithm.

Since D-FOAF/FOAFRealm implements standard authorization mechanisms in application servers, the ACL entries are serialized to string identifiers (see Fig. 3). A simple format represents the root person, minimal trust level and maximal distance from the root person required to access the resource. It is then possible to use such a descrip-

tion in the runtime to specify ACL.

4. 3. Future work

At the moment D-FOAF supports only J2EE platform. We are working on the .NET and PHP support. Since security is a very important to D-FOAF, additional security improvements will be delivered to D-FOAF as well.

To be able to improve the usability of D-FOAF services a ubiquities computing paradigm is being identified and will become a key-feature of the next generation of D-FOAF called DigiMe. We are aiming to give sensitive profile information back to the users' mobile devices and personal computers.

5. Related Work

In the introduction of this article, we described the benefits that an appropriate distributed identity management system can bring to users and applications that interact with web servers and services. We briefly reviewed and compare two commercial initiatives: Microsoft Passport and Sxip.

SUN and another 150 global organizations are leading a parallel initiative called Liberty Alliance Project[7] (http://www.projectliberty.org). This alliance has provided several specifications for an identity federation framework that supports multiple identity providers and a distributed and partitioned store for identity information. Part of this set of specifications is focused in the area of Web Services.

WS-Federation[11] is a new initiative promoted by Microsoft and IBM defines mechanisms that are used to enable identity, account, attribute, authentication, and authorization federation across different trust realms.

Finally, [9] and [1] are two technical reports that provide a good overview of the problem of identity management.

6. Conclusions

We have identified the need for the distributed user profile management system. A number of potential security treats has been presented with a number of possible solutions.

Finally, we have presented D-FOAF systems delivering distributed user profile management. With its strong relationship with RDF and HyperCuP, D-FOAF is unrevealing the potential of Second Generation Internet – the Semantic Web even in currently services completely unaware of that technology. D-FOAF aims to become widely accepted, secure, user-oriented distributed user profile management system.

D-FOAF has been successfully deployed in JeromeDL – e-Library with semantics system[9][5] and MarcOnt collaboration portal[10][6]. The semantically enhanced search features in JeromeDL are strongly dependent on the extensible user profile information delivered by D-FOAF.

Acknowledgement

Authors would like to thank Stefan Decker, Andreas Harth and FOAF community for consultations. The work is supported by SFI (Science Foundation Ireland) under DERI-Lion project (SFI/02/CE1/l131), European Commission 6th Framework Programme – Knowledge Web (FP6-507482) and by KBN (State Committee for Science Research, Poland) under grant 4T11C00525

References

[1] M. Casassa Mont, P. Bramhall, M. Gittler, J. Pato, O. Rees: *Identity Management: a Key e-Business Enabler*. Hewlett-Packard Laboratories, UK. HPL-2002-164 Technical Report

[2] D. Hardt: *Personal Digital Identity Management*. FOAF Workshop, 2004, Galway

[3] W. Josephson, E. G. Sirer, F. B. Schneider: *Peer-to_Peer Authentication with a Distributed Single Sign-On Service*

[4] S. R. Kruk. F*OAFRealm - control your friends' access to resources*. In Proceedings of the 1st Workshop on Friend of a Friend, Social Networking and the (Semantic) Web (FOAF Galway), Galway, Ireland, pages 1–9, September 2004.

[5] S. R. Kruk, S. Decker, and L. Zieborak. *JeromeDL - Managing Digital Library Database with the Semantic Web Technologies.* In Proceedings of DEXA2005, Copenhagen, Denmark, August 2005.

[6] S. R. Kruk, M. Synak, and K. Zimmermann. *MarcOnt — Integration Ontology for Bibliographic Description Formats.* In Proceedings of the International Conference on Dublin Core and Metadata Applications (DC-2005), Madrid, Spain, September 2005.

9 JeromeDL: http://www.jeromedl.org/

10 MarcOnt Initiative: http://www.marcont.org/

[7] Liberty Alliance & WS-Federation: *A Comparative Overview*. White Paper. October 14, 2003. https://www.projectliberty.org/resources/whitepapers/wsfed-liberty-overview-10-13-03.pdf

[8] K. Mitnick, W. L. Simon: *The Art of Deception.*

[9] J. Pato: *Identity Management: Setting Context*. Trusted Systems Laboratory HP Laboratories Cambridge. HPL-2003-72, Technical Report. April 8th , 2003.

[10] M. Schlosser, M. Sintek, S. Decker, W. Nejdl: *HyperCuP - Hypercubes, Ontologies and P2P Networks.* Springer Lecture Notes on Computer Science, Vol. 2530 – Agents and Peer-to-Peer Systems

[11] *Web Services Federation Language (WS-Federation)*. Microsoft White Paper. Version 1.0. July 8 2003 http://msdn.microsoft.com/webservices/community/workshops/default.aspx?pull=/library/en-us/dnglobspec/html/ws-federation.asp

[12] L. Xiong, L. Liu. PeerTrust: *Supporting Reputation-Based Trust in Peer-to-Peer Communities*. To apper in IEEE Transactions on Knowledge and Data Engineering (TKDE), Special Issue on Peer-to-Peer Based Data Management, 2004