Virtual Private Social Networks

Mauro Conti
CS Department
Vrije Universiteit Amsterdam
De Boelelaan 1081a
1081 HV Amsterdam, NL
m.conti@vu.nl

Arbnor Hasani
CS Department
Vrije Universiteit Amsterdam
De Boelelaan 1081a
1081 HV Amsterdam, NL
arbnor.hasani@student.vu.nl

Bruno Crispo
DISI
University of Trento
Via Sommarive 14
38123 Povo, Trento, IT
crispo@disi.unitn.it

ABSTRACT

Social Networking Sites (SNSs) are having a significant impact on the social life of many people—even beyond the millions of people that use them directly. These websites usually allow users to present a profile of themselves through a long list of very detailed information. However, even when such SNSs have advanced privacy policies, users are often not aware of their settings and, on top of that, users cannot abstain from sharing a minimum set of information (e.g. name and location). Such a small set of information has been proven to be enough to completely re-identify a user [22, 25].

In this work we introduce the concept of Virtual Private Social Networks (VPSNs), a concept inspired by the one of Virtual Private Networks in traditional computer networks. We argue that VPSNs can mitigate the privacy issues of SNSs, building private social networks that leverage architecture publicly available for SNSs. Furthermore, we propose FaceVPSN, which is an implementation of VPSNs for Facebook, one of the most used SNSs. FaceVPSN is the first privacy threats mitigation solution that has a light and completely distributed architecture—no coordinator is required. Furthermore, it can be implemented without any particular collaboration from the SNS platform. Finally, experimental evaluation shows that FaceVPSN adds a limited overhead, which, we argue, is acceptable for the user.

Categories and Subject Descriptors

K.4 [**Privacy**]: Abuse and crime involving computers. Intellectual property rights; K.6.5 [**Security**]: Protection

General Terms

Security

Keywords

Virtual Private Social Networks, Internet and Privacy, Social Networking Sites, Facebook Privacy

1. INTRODUCTION

Social Networking Sites (SNSs) are Internet-based applications that allow for user-generated content to be published and accessed easily by a global audience. Some of the SNSs are the most visited sites on the Internet. The biggest SNS, i.e. Facebook, which has more than 500 million users [9], allows users to create an account with an extensive choice of profile information fields (e.g. Basic Information, Interests, Profile Picture, Contact Information, etc.).

The ubiquity of social networks, present in an ever increasing number of people's life, has led to increasingly more private information being "leaked" online. In many cases (i) this is due to just carelessness of users. In many other cases, (ii) information is "leaked" because of the design of privacy controls of the SNSs. As an example, in Facebook, Name, Profile Picture, Gender, and Networks, are considered "publicly available information" (before a recent change [9], even Current City, Friend List and Pages one is a fan of were considered public). That is, such information, according to Facebook may *"be accessed by everyone on the Internet (including people not logged into Facebook), be indexed by third party search engines, and be imported, exported, distributed, and redistributed by Facebook and others without privacy limitations"* [9]. Furthermore, not only can anyone view these fields by simply knowing someone's user ID or username, but also they can be accessed by third party application developers and websites. In other words, a Facebook user cannot abstain from sharing certain information with other third parties. Finally, (iii) inference (of private attributes) can be accomplished by joining data of the profiles the user has in different SNSs (e.g. Facebook, LinkedIn, Twitter, Flickr)—while harder to get, information available from the real world could also be used. Consequently, the privacy related issues have raised the interests of security researchers.

The significance of the privacy issue is escalated by the fact that most of the profiles contain real information [12, 29]. According to [29], in Facebook 99.35% of the users use their real name, 92.2% reveal their birth date, 80.5% their current city, and 98.7% also display a profile picture of themselves. Furthermore, it is relatively easy to re-identify a user that has multiple (public) accounts in various SNSs [22].

There are countless reports on how this amount of data collected from SNSs are being (mis)used. For instance: marketing companies are getting community structure data [8]; identity thieves can perform identity thefts more easily [26]; students or employees are getting in trouble because of information that was found on social networks through de-anonymization techniques [28]; and even the robbers' life has been simplified [2]!

This work addresses these privacy issues by introducing the concept of Virtual Private Social Network to the Social Network environment; this is done in analogy to the concept of Virtual Private Network, VPN, used in computer networks. The main idea is to build private social networks between users leveraging the already existing architecture of SNSs. It is essential for a VPN to benefit from the existing infrastructure. Members of the VPN are network nodes, and they can communicate in a way that is confidential with regards to the nodes outside that VPN. Analogously, a VPSN leverages the infrastructure of an already existing social network, primarily because it is very unlikely for users to move away from popular social networks. Members of the VPSN are members of the social network they leverage. VPSN members can share information (specifically, the attributes of the profile) in a way that is confidential with regards to: i) the other members of the underlying social network; and ii) also against the social network administrator.

We underline that the concept of VPSN is not already present in current SNSs. That is, a user is not able to implement a VPSN with the available tools. While it is not clear whether privacy is a motivation strong enough to justify the implementation of a new SNS from the scratch, we believe the first SNS platform able to offer strong privacy will have a major competitive advantage [7]. However, our focus is on leveraging existing infrastructure, and mitigating the privacy problem for SNSs that already count millions of users.

We focus more specifically on Facebook—the most popular SNS—for which we implement our concept of Virtual Private Social Networks (VPSNs). We call this implementation FaceVPSN. The central problem being addressed is how to use Facebook as a VPSN to share real information only with whom the user intends to—without any other third party (including application developers) being able to see the real information.

Consequently, the aim of FaceVPSN is to implement VPSNs in Facebook. The robustness of FaceVPSN lies in the fact that it is technically compatible with Facebook and it does not require its collaboration, while it completely leverages the already present and (freely) available Facebook architecture and systems—without requiring any additional infrastructure as other solutions do (e.g. FaceCloak [18] and NOYB [13]). In FaceVPSN, the required system resources are distributed among the VPSN members: each member is required to have a small storage and computation overhead that are proportional to the number of members in the VPSN the user belongs to.

Contribution. In this paper we propose and prototype, to the best of our knowledge, the new concept of Virtual Private Social Network with a distributed architecture. The solution does not rely upon the active collaboration of the SNS, i.e. Facebook. The contribution of this paper is threefold: 1) specification of the main security features and properties of a VPSN, and a description on how to realize them; 2) FaceVPSN, the design and implementation of an application, in the form of a Firefox extension, allowing users to implement VPSN in Facebook; 3) an evaluation of FaceVPSN.

Organization. The rest of the paper is organized as follows. Section 2 introduces related work in the field of privacy issues in social networks. Section 3 introduces the new concept of Virtual Private Social Networks (VPSNs). Section 4 presents FaceVPSN, that is our implementation of VPSNs for Facebook. Section 5 contains the evaluation and discussions about the proposed solution. Finally, in Section 6 we present our concluding remarks.

2. RELATED WORK

Following the popularity of social networks, the issue of privacy in social networks has gained momentum in the last couple of years. Nonetheless, most of the published research deals primarily with general de-anonymization, and in few cases with anonymization techniques that are not designated for a specific SNS. We shall also bring to the reader's attention the fact that most of these works, referring to Facebook, have been made obsolete by the latest changes (in May 2010) of the Privacy Policy of Facebook [10].

Many papers investigate the different facets of privacy in SNSs. The authors in [19], [30], and [14] emphasize the issue of communities centered around attributes (i.e. female computer scientists in Amsterdam) when approaching privacy problems. They show the possibility to predict, with high probability, a user's attributes using the attributes of her friends and contacts. [15] presents a privacy protocol allowing anonymuous opinion exchanges over untrusted SNSs. In [16], the authors investigate the number of user accounts needed to be compromised to rebuild the entire social graph. This number is strictly dependent on the SNS's lookahead (a social network has lookahead l if a user can see all of the edges incident to the nodes within distance l from him). The paper recommends to limit the lookahead to 1 or 2, for any social network that wishes to decrease its vulnerability.

About solutions, in [3] the authors proposed a user privacy policy (in the form of a contract) to ensure the proper protection of private data. SNSs are supposed to implement such policies to improve the privacy of their users. Despite a prototype implementation being presented in [4], to have an impact the solution requires existing SNSs to adopt such a policy. Similarly, Persona [5] is a social network in which the user defines a privacy policy. This policy manages access to user information. Persona uses an application named Storage to allow users to store personal data, as well as share them through an API with others. The authors have integrated Persona with Facebook as an application to which users log-in through a Firefox extension, which interprets the special markup language used by Persona applications. Only users of Persona, with necessary keys and access rights, will be able to access the data from Storage, and view them in their browser. Nonetheless, it is only another Facebook application that can easily be removed by Facebook from the applications directory.

Another interesting recent thread of research is considering the possibility of new competitor social networks designed with privacy in mind [7, 27]. We expect those solutions, when supported by a good business model, to reach a wide consensus from users. However, differently from these solutions, our aim is to mitigate threats to the privacy of millions of users that are currently using existing SNSs. That is, we aim at leveraging (for free) the existing SNSs infrastructure to build on top of them Virtual Private Social Networks (VPSNs).

In [11], the authors propose the "privacy by proxy" API to allow Facebook to reveal users' private information only if this is explicitly allowed by the user. This is done by interpreting special FBML (Facebook Markup Language) tags (i.e. Facebook specific HTML-like tags) in the output of the applications. However, the solution in [11] requires the cooperation of Facebook. Similarly, another proposal, flyByNight application [17], requires the cooperation of Facebook. flyByNight provides secure messaging between Facebook friends, by encrypting and decrypting sensitive data using client-side JavaScript. However, it does not solve the privacy concerns with regards to publicly available information.

Another solution for privacy of communication in SNSs is the Secret Interest Groups (SIGs) framework [24]. Although, not a direct solution to privacy of public information, it offers an interesting new way to create self managed "secret groups" (about private topics), and control group memberships. The idea is integrated with Facebook as an implemented Java HTTP proxy. While the concept of SIGs is close to the one of VPSNs, we note that the solution suggested in [24]—even if not centralized or having a single authorization entity—envisages the presence of one or more managers that, for example, allow new users to join the SIG. Furthermore, even for the managers, any operation is "thresholdized". That is, more managers need to join and agree in order to make a decision (e.g. add a new member). We underline that such an approach is completely different from the one we propose in our work. That is, each single user is completely independent on defining who (and how) will be able to access his private information. Finally, for all the solutions implemented through a proxy, we argue that a user might not be willing to have a proxy filtering all of his browsing.

More similar to our approach, is the one of FaceCloak [18]. FaceCloak is a Firefox extension allowing users to specify which data in their profile, or Facebook activity, need to be kept private. The sensitive data are substituted with fake ones, while the encrypted version of the private data are published on a third party server. Users email keys to each other, so that their respective FaceCloak extensions can retrieve the real text from the third party server. Then, each time a user retrieves a friend's profile from Facebook, in order to substitute the fake data with the real ones FaceCloak must retrieve also the encrypted version of the sensitive data from the third party server. Hence, FaceCloak relies upon the availability of a "parallel" centralized infrastructure to store the encrypted data rather than leveraging on the existing Facebook platform. An obvious drawback of this solution is that users have to trust the security of the third party server. Also, the third party server represents a single point of failure.

NOYB ("None Of Your Business") [13] is another system for privacy protection on Facebook. Private profile information is divided into atoms (e.g. name and gender constitute one atom), and each atom is "replaced" with a corresponding atom from another randomly selected user who uses NOYB. Only authorized users can reverse the process of attributing the atoms of information to the real profile. NOYB is implemented as a browser add-on for the Firefox web browser. The problem with NOYB is that the anonymity depends on the number of its users.

None of the work in literature mention the concept of Virtual Private Social Networks (VPSNs). Furthermore, most of the reviewed literature does not address the problem of publicly available information on Facebook. The few proposed solutions that we mentioned above either (i) rely upon an additional centralized infrastructure or (ii) rely upon the collaboration of Facebook and add unnecessary encryption overhead in users' interaction with Facebook. In contrast, the solution proposed in this paper does not suffer from these limitations.

3. VIRTUAL PRIVATE SOCIAL NETWORKS

In this section, we introduce the concept of Virtual Private Social Networks (VPSNs), and describe their security features and properties. The concept of VPSNs is analogous to the concept of Virtual Private Networks (VPNs) applied to traditional computer networks. The mapping between VPN and VPSN can be seen as follows. First, both VPN and VPSN leverage an already pre-existing infrastructure: a real network in the case of VPN and a real social network in the case of VPSN. We will also refer to the already existing network as the host social network, in case of a VPSN. A network node in a VPN is mapped to a user in the VPSN. Nodes of a VPN can communicate in a way that is confidential with regards to other nodes outside the VPN. Similarly, in a VPSN, users can share information that are confidential with regards to other users which are not part of the VPSN. In particular, the information shared by users of a VPSN are those related to the user's profiles (e.g. Name, Profile Picture, and Current City).

We expect a VPSN to enjoy the following features:

- **Virtual.** A VPSN should not have its own infrastructure. Instead, a VPSN has to leverage the architecture and the infrastructure of an already existing social network.

- **Private.** The users that are not part of the VPSN should not be able to access profile information shared within the VPSN. This information must be confidential even with regard to the host social network manager (e.g. Facebook).

- **A social network.** The VPSN build on top of a real social network (e.g. Facebook) should remain a social network itself (as defined in [6] and extended in [7]). That is, it should have the same functionalities of the social network it leverages, even if the shared information is restricted to the VPSN member. In particular, we do not expect the host social network to be used only as a repository, just as we do not expect the VPSN actually being a complete social network infrastructure.

- **Hidden to users outside the VPSN.** The users that are not part of a VPSN should not be able to know about the existence of the VPSN (hence not even the VPSN members can be identified as so).

- **Hidden to the host social network manager.** SNSs managers might not encourage the use of VPSNs, because of the reasons exposed in the Introduction. For example, VPSNs would reduce the benefit of targeted advertisement (e.g. directed to profiles claiming a given current city). Hence, we envisage VPSNs to be hidden to the SNSs managers. In particular, VPSNs should not be identifiable by the SNSs, and of course they should not depend on the SNSs collaboration.

- **Transparent.** Once within the VPSN, the users should be able to use the service in a transparent way. That is, a user should be able to browse the real profile of a friend, as it would happen if they were both on the host social network, and sharing there the real profile information.

- **User might connect to more than one VPSN.** Similarly as it happens for VPNs, in VPSN we expect to be able to partake in several VPSNs. In particular, the user might: i) appear in two (or more) VPSNs with different profiles (e.g. appearing like currently being in Amsterdam in the first VPSN, while being in Rome in the second VPSN); ii) appear in different VPSNs with different levels of privacy (allowing friends in the first VPSN to see his current city, while retaining this information from friends in the second VPSN).

- **View.** We also extend the previous concept of a user being connected to different VPSNs, allowing even more fine tuning within the same network. In fact, the concept of appearing with different profiles to different sets of people can be generalized—a user might appear with a different profile (or with a different level of availability of information) to each single friend.

- **View evolution.** The view that one user has of another user's profile might evolve. That is, the amount and type of information shared might change over time.

- **Light.** The infrastructure required to let the VPSN work should be negligible and not be comparable to the one of the host social network. The induced overhead in terms of memory and computation should be as trivial as possible.

We observe that VPSNs cannot be easily implemented with the tools currently available in SNSs. One thing that might resemble a VPSN are the Facebook "groups" (or similar concept for other SNSs). However, a group is known to the SNS and it depends on the SNS collaboration. Even if we ignore this fact, there are other two major differences between a "group" and a VPSN. First, groups are centrally managed while VPSN management is completely distributed. Second, groups have a simple access control rule. Once part of the group, all the information within the group are available to the new member. On the other hand, with VPSN it is possible to create different "views" inside the same VPSN.

We underline that the focus of this work is about the privacy of (public) information shared as part of a social network profile. Other aspects connected to the users identification (e.g. traffic analysis), not based on this information, are orthogonal to the scope of this paper.

4. VPSNS IN FACEBOOK: FACEVPSN

In this section, we present FaceVPSN, which is our VPSN implementation for Facebook. To the best of our knowledge, this is also the first solution to address the problem of availability of public information—not only with respect to other users, but also against third party applications in Facebook. The concern for the availability of this information to third party application is confirmed by the fact that Facebook recently (May, 2010 [9]) gave the user the possibility to completely turn off the applications platform (before these changes, third party applications were able to access by default user's "publicly available information" that consisted of Name, Profile Picture, Gender, Current City, Networks, Friend List, and Pages). Nevertheless, if a user has not turned off the applications platform, Name, Profile Picture, Gender, Networks, and User ID are still publicly available by default. The availability of this option is undermined by the evidence that *"every month, more than 70% of Facebook users engage with platform applications"* [9]. Hence, users do not just forget to turn off applications platform, rather they explicitly want and use applications. We consider 70% of Facebook users to be a high enough number for the privacy problem to be considered an unresolved issue. In addition, the availability of profile information to applications raises an even higher concern. In fact, applications can easily retrieve and process a big amount of profile information. Finally, we note that even if users choose to turn off applications platform, public pieces of information are still available just by browsing profiles, e.g. with `USER_ID` (`www.facebook.com/index.php?profile=USER_ID`), not mentioning the risk [23] of data stored by Facebook being hacked.

An overview of our solution, named FaceVPSN, is presented in Section 4.1. The architecture is presented in Section 4.2, whereas the implementation details are discussed in Section 4.3.

4.1 Overview

The simple solution of encrypting the attributes would not work, since Facebook does not allow publication of ciphertext or even scrambled text. Our approach is the following: we cannot stop the sharing of public information due to the design of Facebook, however we can publish pseudo information, which inevitably can be seen by third party applications. According to [29], this is one of the less frequently used privacy protection strategies by Facebook users. This behaviour is actually motivated by the following need: allowing the Facebook friends of a user to see his real Name, Profile Picture, Gender and Current City.

Given that User ID, Network names, Friend's names, and the names of the Pages a user is a fan of cannot be changed by the user in any way, the real concern is with regards to Name, Profile Picture, Gender and Current City. Hence, we propose that those fields are modified, through the Profile Editor of Facebook, in order not to contain the real values that correspond to Facebook account owner: we suggest to use pseudo values instead. This modification should be done by the user himself for three reasons:

1. The user can select the pseudo information to display (namely Name, Profile Picture and Current City), and thus still be in control of his published profile page (e.g. how his pseudo profile picture will look);

2. The user can choose not to share some of the real details with certain friends;

3. Users are already familiar with modifying profile information, hence doing this modification is not a barrier for using our proposed system.

Once pseudo information is stored on Facebook, the real information is sent only to friends allowed to see it. This information will be stored locally on the friends' machines, with a mechanism that is similar to how cookies work in browsing. However, differently from cookies, once this information is stored locally, it will not be sent back on a later occasion. Instead, it will be processed locally, when required. When a user browses a profile of another user in the VPSN, a FaceVPSN component is in charge to transparently show to the user the real information, instead of the one actually published on the host social network.

A possible variant to our current implementation might be the following. Instead of storing information locally, it might be stored steganographed within images attached in Facebook messages. FaceVPSN would take care of transparently managing this information when required. This approach would make the solution independent from the specific machine where the real information is stored. In this way, the user will be free to browse the VPSN from each machine where he is able to log-in on Facebook and install FaceVPSN.

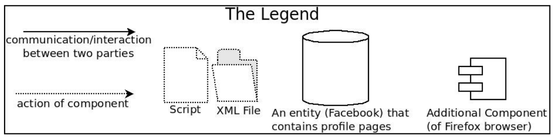
4.2 Architecture

We implemented the replacement of pseudo information with real one through a Firefox extension. This is done whenever pseudo information is loaded on the user's Firefox browser. The Firefox extension we have implemented is called FaceVPSN and it replaces Name, Profile Picture, and Current City. The application is designed in such a way that it is also easily extendable to handle the replacement of other profile information. We use the notation in Figure 1(a) to illustrate the FaceVPSN architecture (Figure 1(b)).

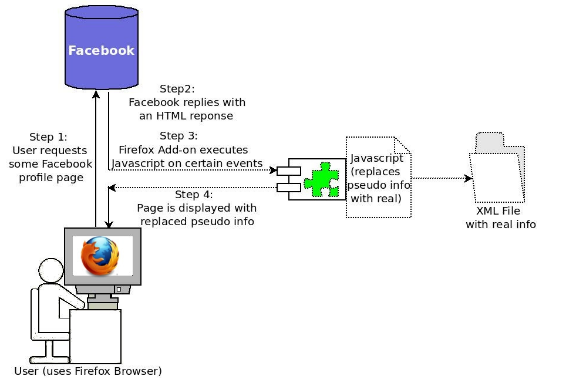
After a Facebook page is requested (Step 1 in Figure 1(b)), Facebook replies with an HTML response that contains the content of the page to be displayed by the Firefox browser (Step 2). The functionality of FaceVPSN is implemented in JavaScript code, which is executed once certain events happen. In particular, when the document is loaded from Facebook, a "page load" event is fired (Step 3). Then, the JavaScript code of FaceVPSN searches to find pseudo names, cities, and profile pictures of the friends of the user. Afterwards, the JavaScript code replaces them with the real information stored in XML files. In the end, the Firefox browser displays the profile page with replaced pseudo information (Step 4).

In order to partake in the VPSN, the user needs to set up FaceVPSN, which currently works only with Mozilla Firefox web browser. A general overview of the required interaction for FaceVPSN set up is given in Figure 1(c) (using the notation in Figure 1(a)).

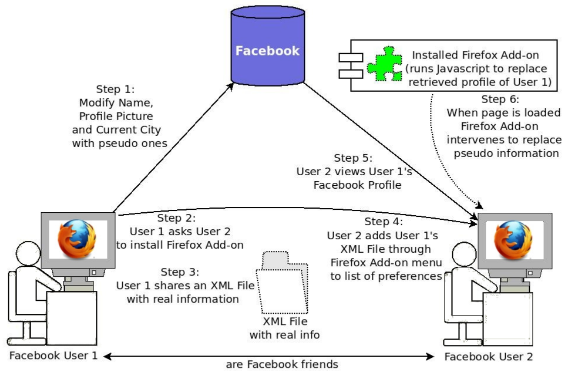
In Step 1, Facebook User 1 would modify his Name, Profile Picture, and Current City to some pseudo information through the profile editing interface of Facebook. Next, User 1 asks one of his friends, say, Facebook User 2, to install FaceVPSN and also emails him an XML file that contains details of pseudo and real information. User 2 would install FaceVPSN just like any other Firefox extension. It is as simple as dragging the downloaded XPI installation file to Firefox browser and a menu will pop up to confirm installation. User 2 has to add the XML file corresponding to User 1 to the list of preferences. This is done through the Graphical User Interface (GUI) of FaceVPSN. All that remains is for FaceVPSN to replace the pseudo information of Facebook User 1 whenever User 2 views User 1's profile, or a page that contains User 1's information (Step 5 and 6 in Figure 1(c)).


The Legend
communication/interaction between two parties
action of component
Script
XML File
An entity (Facebook) that contains profile pages
Additional Component (of Firefox browser)

(a) Legend for figures 1(b) and 1(c).


Facebook
Step2: Facebook replies with an HTML reponse
Step 1: User requests some Facebook profile page
Step 3: Firefox Add-on executes Javascript on certain events
Step 4: Page is displayed with replaced pseudo info
Javascript (replaces pseudo info with real)
XML File with real info
User (uses Firefox Browser)

(b) Browsing


Facebook
Installed Firefox Add-on (runs Javascript to replace retrieved profile of User 1)
Step 1: Modify Name, Profile Picture and Current City with pseudo ones
Step 6: When page is loaded Firefox Add-on intervenes to replace pseudo information
Step 5: User 2 views User 1's Facebook Profile
Step 2: User 1 asks User 2 to install Firefox Add-on
Step 4: User 2 adds User 1's XML File through Firefox Add-on menu to list of preferences
Step 3: User 1 shares an XML File with real information
XML File with real info
Facebook User 1
are Facebook friends
Facebook User 2

(c) Setup

Figure 1: FaceVPSN Architecture.

4.3 Implementation

FaceVPSN has been implemented as a Firefox extension and its functionality is written in JavaScript. The GUI has been written with XUL, the Firefox XML user interface markup language. In order to share real information in a structured way, that will be useful and easily accessible by the Firefox extension, we propose to use an XML file with the structure of the example shown in Figure 2. The user is not required to fill in every field. Nevertheless, he must specify at least his username and user ID. Then, for instance, he can choose to have only pseudo Name and City but no Profile Picture.

Essentially, everything is contained under the root tag `<FaceVPSN>`. We defined the current version to be *"v0.1"* in order to track future changes of structure of the file. In Figure 2 we show the typical case, that is, when a user shares only his information. Nonetheless, it is also possible that two or more users can share information in the same XML file; hence using more `<friend>` tags—the information of one person is contained inside those tags. We believe that the choice of using this implementation is convenient, for instance, when only one of close friends or siblings has to

```
<FaceVPSN version="0.1">
 <friend>
  <urls>
    <username>facebook.username</username>
    <id>user-ID-00000</id>
  </urls>
  <replacementinfo>
    <info>
      <pseudoname>"Pseudo Name"</pseudoname>
      <realname>"Real Name"</realname>
      <pseudocity>"Pseudo Current City"</pseudocity>
      <realcity>"Real Current City"</realcity>
      <pseudopicurl>"http://pseudoSmall.jpg"</pseudopicurl>
      <realpicurl>"http://realSmall.jpg"</realpicurl>
      <pseudosearchpicurl>"http://pseudoS.jpg"</pseudosearchpicurl>
      <realsearchpicurl>"http:/realS.jpg"</realsearchpicurl>
      <pseudobigpicurl>"http://pseudoBig.jpg"</pseudobigpicurl>
      <realbigpicurl>"http://realBig.jpg"</realbigpicurl>
    </info>
  </replacementinfo>
 </friend>
</FaceVPSN>
```

Figure 2: Example or XML File Structure.

take care of sharing the file for two or more people. Moving down the XML hierarchy, the reader can observe that we classified information into two categories:

- information contained inside `<urls>` tags, which will be used to check URLs to verify to whom the link or profile page corresponds to. This includes user's Facebook username (contained inside `<username>` tags) and user ID (contained inside `<id>` tags);
- information that will be used to replace pseudo information, and contained inside `<replacementinfo>` tags. This category of information contains the pseudo Name and Current City, and the corresponding real values. Additionally, there are three different Profile Picture URLs in Facebook. The most frequently used is the small boxed one, that appears for example in Wall posts. Yet, there is also a bigger profile picture that appears in a user's profile page with a different URL. Finally, there is another one that appears in search results.

Users add on FaceVPSN the XML files that refer to their friends—in order for FaceVPSN to know whose details to replace. FaceVPSN uses the preferences system of Mozilla [20] in order to save the user's preferences when the browser is started and closed. It also uses the input stream component of Mozilla [20] to actually open and read a file that the user has chosen from the dialog window. FaceVPSN handles replacement of pseudo names in text as well as links. Furthermore, it replaces the real name with a pseudo name when searching in Facebook (e.g. when tagging or in the search bar). Current City is replaced by checking the URL in the address bar to find whose profile is being viewed, to know to whom does a current city correspond to. The document's URL is checked to find either the friend's ID or username. The URL though, can contain a hash "#" that specifies a named anchor in the document. If that is the case, then there will be two different user IDs or usernames in the URL. Consequently, if the hash "#" is present in the URL, FaceVPSN always checks the part after it for each friend's username or user ID. On the other hand, profile pictures are replaced by changing image source to a different source link (remote or local).

To start the execution of replacement functions we add a Gecko (which is the Firefox rendering engine) specific event (i.e. DOMContentLoaded). This event is fired when the DOM Content of a page is loaded, albeit without any documents or images on it. Nonetheless, when the user navigates in Facebook, FaceVPSN has to re-perform replacements due to new content being displayed. For instance, the URL changes when a new picture of an album is being viewed or a different profile tab is opened. To get notified whenever the URL in the address bar changes, we added a progress listener to our code to re-apply replacements whenever location changes. Progress Listeners [20] allow Firefox extensions to be notified of events associated with documents loading on the browser and with tab switching events. More details on the implementation, are available on the project website [1].

In order to have the published (non real) information replaced on the browsed page, FaceVPSN has to match different regular expressions to find them. These are used primarily to find information that needs to be replaced, but also, for instance, if we consider the case of two friends with the same name then the replacements are to be performed based on some regular expression matching. This requirement can generally be fulfilled by distinguishing information based on usernames or user IDs, which are unique. Nevertheless, there is also a possibility to exactly match the places where information (i.e. we are referring to names here) is to be substituted. In this context, we could construct a number of regular expressions that would allow FaceVPSN to find and replace information in exact specified links (e.g. Name appearing in search list, Name appearing in View Photos of Friend link). In total we could come up with as far as 30 regular expressions (the list of regular expression used can be found in the Appendix).

Of course, the defined regular expressions are SNS dependent. That is, if the Facebook implementation change they must be adapted. For this, we envisage FaceVPSN to be continuously supported (e.g. by the open source community) and to automatically check for updates. Similarly, the regular expressions must be adapted if we aim at designing a VPSN implementation for other SNSs (e.g. Orkut), while the same general concept can be applied.

5. EVALUATION AND DISCUSSION

The implementation of FaceVPSN shows that it is indeed feasible to actualize the concept of VPSN. However, in this section we also give a thorough evaluation and discussion of FaceVPSN. First, we compare FaceVPSN with other solutions in literature (Section 5.1). Given that, to the best of our knowledge, the concept of Virtual Private Social Network is new to the literature (hence, there is no implementation of VPSN), we compare FaceVPSN with other solutions for mitigation of privacy threats in Facebook. Further on, in Section 5.2 we discuss the results of experimental evaluation we performed with FaceVPSN.

5.1 Comparison

Unlike other solutions, FaceVPSN is a decentralized solution that does not depend on the collaboration of Facebook. Moreover, handles the replacement of the "publicly available information" that the user can modify (i.e. Name, Profile Picture, and Current City). In particular, FaceVPSN enjoys all at once a set of features that no other current solution has. Table 1 summarizes the main points we highlighted with regards to the relevant solutions (about mitigating privacy threats in Facebook) in addition to FaceVPSN. The column headers shows five different features, while row headings indicate the considered solutions. In particular, we compare FaceVPSN with: FaceCloak [18], flyByNight [17], NOYB [13], Persona [5], and SIGs [24]. These are all solutions directly related to Facebook privacy issues. We remind the reader that our aim is to mitigate the privacy threat of the millions of users using SNSs like Facebook, and building VPSN without having an infrastructure comparable to those of SNSs. Hence, we do not consider solutions like Safebook [7] and PeerSoN [27]. In fact, these solutions do not mitigate privacy in Facebook but rather propose new competitor SNSs.

The first column of Table 1 (i.e. Facebook Collaboration) indicates if the solution requires the Facebook collaboration in order to work or exist. More specifically, we are interested to know whether Facebook can remove the application from the interaction cycle of users with Facebook. For instance, the solutions flyByNight, Persona, and SIGs require Facebook collaboration. That is, they can be removed by Facebook from the applications directory and consequently cannot be used by Facebook users. As such, the successful solution of the privacy problems by these applications is under direct threat from Facebook itself. However, considering that potentially every implementation aimed at solving privacy issues in Facebook depends on the Facebook implementation, we do not bring this second viewpoint into this column. For instance, if Facebook adds new profile information fields, NOYB has to be extended to support them or, when Facebook API changes, Persona and flyByNight have to be harmonized accordingly. Likewise, when Facebook changes the design of profile pages and how profile information is organized and displayed (e.g. different HTML tags) then FaceVPSN also needs to be adjusted to handle the changes. Nevertheless, we do not consider this aspect a barrier for the existence of a solution.

We also compared the architecture of the solutions (second column of Table 1). In particular, we observed whether the solution was implemented in a centralized or distributed architecture. In a centralized architecture there is one central component which is fundamental to the functionality of the whole system. This type of architecture suffers from the "single point of failure" problem. Conversely, a distributed architecture is one where there is no need for a central component or coordinator for the operation of the system. A third type, that is "thresholdized", applies to SIGs [24]. In particular, in SIGs a group is managed by a set of so-called managers that take decisions (e.g. admitting a new user in the group) using a threshold consensus. That is, before taking a decision, at least a given number (threshold) of managers have to agree on that decision. FaceCloak, that is implemented as a centralized architecture, requires an infrastructure that becomes comparable to that of Facebook (or even worse, when users belong to many VPSNs). Furthermore, in FaceCloak, if the third party server for storing encrypted private data is down, users' FaceCloak extension cannot decrypt information from Facebook. Similarly, if the flyByNight application server is down, users cannot see decrypted messages from their Facebook friends. In its current implementation, NOYB is also centralized in nature, given that the NOYB Firefox extension depends on external public dictionaries that it queries to retrieve and replace "atoms". Persona as a social network proposal is a decentralized architecture: there is no central coordinator and users themselves define policies for sharing information through their "Storage". Nonetheless, the Persona Facebook application, just like flyByNight, depends on a third party server (i.e. the server where the application resides). In general, the centralized architectures with a single point of failure are potentially a barrier to widespread usage in social networks. On the contrary, FaceVPSN is implemented as a Firefox extension that revolves around its users. As it was specified in Section 4.2, the user himself is in charge of when and what information to share with others. Moreover, if the FaceVPSN Firefox extension of a user fails, this will not affect his friends.

The column labeled as "Overhead" shows the time overhead of the different solutions. To put it differently, if the data is available, it shows the additional time required for a user to see the benefits of the solution. As for previous solutions, the time overhead indicated in Table 1 is the one reported in the corresponding publication paper [18, 17, 13, 5, 24]. The overhead in flyByNight, which handles the encryption of private messages in Facebook rather than "publicly available information", depends on the size of the message. Both [24] and [13] do not provide information about their overhead, while we argue this is a relevant characteristic of these solutions. NOYB has an overhead that depends on the number of atoms to be replaced, as well as the number of different keys that need to be generated. In turn, this depends on the updates of profile fields and is upper bound by the number of fields. As a distinguishing factor of FaceVPSN from centralized solutions, we formally define the storage overhead of FaceVPSN (distributed) compared with the one of FaceCloak (centralized)—a similar comparison holds for other centralized solutions. Let us define: N as the total number of users; u_i the generic user ($1 \leq i \leq N$); the set of VPSNs the user u_i belongs as $V(u_i) = v_1, \ldots, v_j, \ldots v_V$; $Size(v_j)$ as the number of elements in VPSN v_j. We remind the reader that each user can give a different view (i.e. show a different profile) for every other user in the VPSNs he belongs to. FaceCloak [18] requires a storage that is $O(\sum_{i=0}^{N} \sum_{j=0}^{V(u_i)} Size(v_j))$. On the other hand, FaceVPSN (with decentralized architecture) requires a limited local storage (and computation) overhead that is $O(\sum_{j=1}^{V(u_i)} Size(v_j))$, for user u_i.

In the "Pre-requisites" column, we are mainly interested in potential barriers to the usage of the system. For instance, there is no noteworthy barrier in using FaceCloak, unless we take into account the future possibility to pay for a third party server. However, users have to email keys that are used by the system for decryption. With flyByNight though, users always have to use this Facebook application to view the decrypted text of their private messages. The same is true for Persona. Similarly, NOYB needs to be used not only by the user himself, but also by other users, in order to create a richer public dictionary, hence having more possibilities to substitute the atoms of his attributes. Finally, SIGs solution requires the user to browse through a proxy. On the contrary, with our solution the user does not depend on whether his friends choose to use FaceVPSN. However, the user still has to send by email to his friends the XML files with his details—and it is up to his friends to decide if they want to install the FaceVPSN extension and view his real details. We are aware that the need to email and manage XML

	Facebook Collaboration	*Centralized/ Distributed*	*Overhead*	*Pre-requisites*	*Hide public info*
FaceCloak [18]	No	Centralized	1 s (average) to view real text	Manage keys	Partially
flyByNight [17]	Yes	Centralized	depends on message size	Always need to use this Facebook app	Partially
NOYB [13]	No	Centralized	no data	Friends need to use this Facebook app	Partially
Persona [5]	Yes	Distributed	2.3 s (median; max 13.7 s)	Always need to use this Facebook app	No
SIGs [24]	Yes	"thresholdized"	no data	Browsing through the proposed proxy	No
FaceVPSN	**No**	**Distributed**	**<1 s (average) to view initial replacements**	**Manage XML files**	**Yes**

Table 1: Comparing FaceVPSN with state of the art solutions for privacy threat mitigation

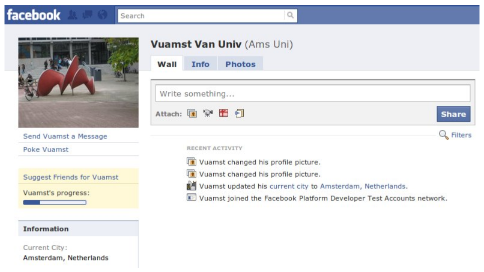
files could hinder the usage of FaceVPSN. However, this is just how it works in the current implementation—ongoing work aims at automatizing these communications. For example, we could leverage the messaging system of Facebook itself (email and Jabber chat) to share XML info. Also, this could be done in a way that is private to Facebook, e.g. steganographed in pictures attached to emails.

Finally, and most significantly, we reviewed each solution to see if it addresses the problem of "publicly available information" (e.g. does it hide it from Facebook and third party applications?). This comparison is in the last column of Table 1, which indicates if the corresponding solution in the row is able to hide the information that Facebook classifies as public (Name, Profile Picture, and Current City), no matter the privacy configuration set by the user. FaceCloak, flyByNight and NOYB hide only partially those information. In particular, FaceCloak and NOYB are able to hide only text data and not pictures, while flyByNight can hide only message data. Persona does not handle profile information hiding at all. In contrast, FaceVPSN allows users to hide their real Name, Profile Picture, and Current City.

5.2 Experimental Evaluation

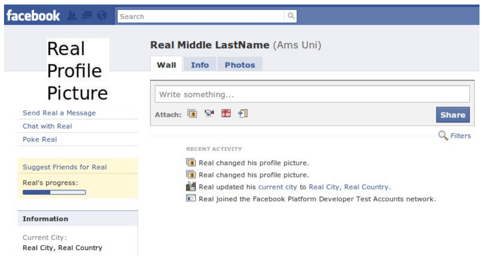
To illustrate the actual work of FaceVPSN, we used a test Facebook account, and created an XML file with the corresponding pseudo details and real details. The published profile resulting from this test account with pseudo information is shown in Figure 3(a). Having FaceVPSN working, and the proper XML file configured, we observed the proper replacement of information in news feeds, pop-ups, and profile pages. The resulting profile page shown to the user with FaceVPSN in action can be seen in Figure 3(b). Besides the profile page, other elements of Facebook are also handled by FaceVPSN. Figures 4(a), 4(c), 4(d), and 4(b) show example replacements in a chat window, notification window, pop-up window, and search bar, respectively.

Unfortunately, FaceVPSN (and privacy in general) comes at a price. To be more specific, the usage of FaceVPSN would deprive a user from some of the functionalitites of the Facebook platform. A user appearing with a pseudo name in Facebook might not get invited to particular events. In fact, this name is not accessible to third party applications through the APIs of Facebook (although this information is visible if an adversary opens the user page after knowing the published user ID). Thus, if users are concerned that their real friends will not be able to find them, they can also choose to display an alternate name in addition to the



Vuamst changed his profile picture.
Vuamst changed his profile picture.
Vuamst updated his current city to Amsterdam, Netherlands.
Vuamst joined the Facebook Platform Developer Test Accounts network.
Amsterdam, Netherlands

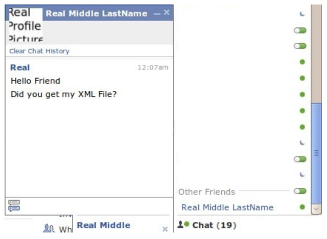
(a) Test account before using FaceVPSN



Real changed his profile picture.
Real changed his profile picture.
Real updated his current city to Real City, Real Country.
Real joined the Facebook Platform Developer Test Accounts network.
Real City, Real Country

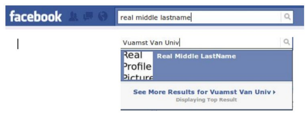
(b) Test account after using FaceVPSN

Figure 3: FaceVPSN in action. Profile Page.

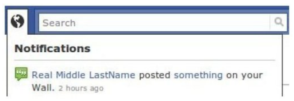


Real 12:07am
Hello Friend
Did you get my XML File?

(a) Chat

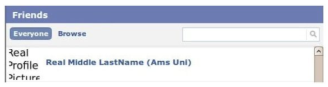


(b) Search Bar



Real Middle LastName posted something on your Wall. 2 hours ago

(c) Notification



(d) Pop-up

Figure 4: FaceVPSN in action in different Facebook features. Name and Profile Picture always replaced as needed.

pseudo one. Furthermore, when a user wishes to change his real information (e.g. Current City), in our current implementation, the user has to re-send the XML file to his friends in order for them to view the updated real information. Ongoing work aims at making the sending and update of XML on the friend's machine more transparent, e.g. leveraging the Facebook messaging services themselves, as discussed in Section 4.1.

In our running FaceVPSN system, we observed that after managing the XML files, the other main issue was the time overhead for replacing the pseudo information with real one. In particular, we found two main variables influencing time performances: (i) the number of friends the user has; (ii) the required level of precision for the pseudo-to-real information replacement. In particular, this level corresponds to the number of regular expressions used (see Section 4.3).

Based on these observations, we ran a thorough set of experiments to investigate how this time overhead varies with the number of friends and considered regular expressions. For evaluation purposes, we used JavaScript Debugger (Venkman [21]) of Mozilla to get some profiling data of execution time of various functions of FaceVPSN. The experiments have been performed on a laptop with 1.86GHz Intel Pentium CPU and with Windows XP Professional with SP3 Operating System. Mozilla Firefox version 3.6.8 was used to conduct evaluation tests during which Firefox was ran with FaceVPSN and Venkman extensions. Besides Firefox, no other user applications were running.

First, we investigated the influence of the number of friends on the time overhead. We considered the browsing sequence path (e.g. browsing the Facebook home page, then the user profile page, and so on) described by the sequence in Table 4. In this experiment we assumed the worst case in terms of possible replacements. That is, we considered at the same time: (i) all the implemented regular expressions (refer to tables 2 and 3), and (ii) all the replacements running automatically (Table 4). We also assumed the worst case in terms of the number of user's friends that are within the VPSN of the user. That is, we assume FaceVPSN has to replace information for all the friends of the user. We measured the time overhead for the given browsing sequence path, for 1, 3, 5, 10, 15, 20, and 25 friends. For each number of friends, we run the browsing sequence 10 times (each sequence being made of the 20 steps described in Table 4), and generated the sample reports with data about execution times of FaceVPSN functions. The measured values represent the time when all the replacements are performed on a page, rather than how long afterwards will the first replacements be seen. The results obtained from this experiment are summarized in Figure 5(a). For each point on the x-axis (that is, for each given number of friends), the corresponding value on y-axis shows the average time overhead in a single step of the browsing sequence.

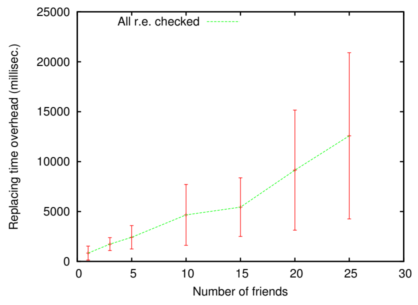
As expected, with the increase of the number of friends, whose details need to be replaced, there is an increase in the execution time. For example, the average replacing time overhead is 2,421 milliseconds for 5 friends in the user's VPSN, and 12,585 for 25 friends in the user's VPSN. This behaviour is understandable when considering the fact that replacements are performed many times for the various friends in the news feed, and also when HTTP requests are generated or URL changes in the address bar. We recall that FaceVPSN performs name replacement in various links (e.g. Friend's First and Last Name in profile page posts, someone's Photos, someone's Links etc.) through regular expressions. In particular, this experiment was ran considering all the implemented regular expressions. We implemented these regular expressions in a way such that we were sure to handle each and every possible occurrence or required replacement, and also not to make any mistake. For example, to handle cases of friends with the same name, it is required to check with every friend's username and user ID.

Our first intention was to design a solution that, in terms of available features, was completely transparent if compared to the standard behaviour of Facebook. While the described full-featured implementation reached this aim, we now observe two issues. First, for a user even not handling all the cases might still be a reasonable behaviour. For instance, it is not essential to perform (automatic) replacement of pseudo information when user wants to view all comments in a post. This is because user has already seen the real information in some of the comments that are displayed initially. Furthermore, we observed that some of the browsing steps (steps 8, 13, and 16 in Table 4) differentiate from the others for the particular high time overhead they require. These operations are used to handle Ajax requests that, for example, implies continuous replacement while user is typing a name in a search bar. For example, let us assume the user wants to search for a friend with real name "Name". After user types just "N", a reply (for the Ajax request) with a suggestions list will show the list of all the friends that have a name starting with "N". After the user types "Na" the list will show all friends' names starting with "Na", and so on.

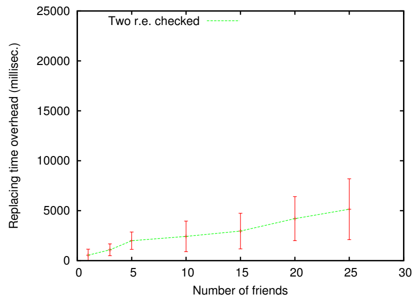
To investigate the influence of these two issues, we run further experiments. First, we wanted to study the influence that the number of considered regular expressions has on the time overhead. In particular, we re-ran the previous experiment only with two regular expressions being matched (i.e. the first two in the Table 3 in the Appendix). One is used to match a friend's username, whereas the other one is used to match user's ID. The results of this experiment are reported in Figure 5(b). Comparing this figure with the previous one (Figure 5(a), which shows results for all the implemented regular expressions), we can see that the time required for the replacement is significantly reduced. For example, considering 25 friends whose information need to be replaced, the replacing time overhead decreases from an average of 12,585 milliseconds to 5,145 milliseconds.

We can observe that the implementation of FaceVPSN with many regular expressions increases the overhead to an undesirable level. However, we shall point out that the reason we constructed all the regular expressions is that we wanted to be assured we are replacing names only in places we can undoubtedly say that they refer to a particular friend or user. The implementation with only two regular expressions (i.e. to match usernames and user IDs) is more efficient (refer to Figure 5(b)). Also, based on our tests, FaceVPSN (with two regular expressions) manages to match and replace all the names correctly. Hence, we argue that with two regular expressions we can get a reasonable level of experience with correctly substituted pseudo information.

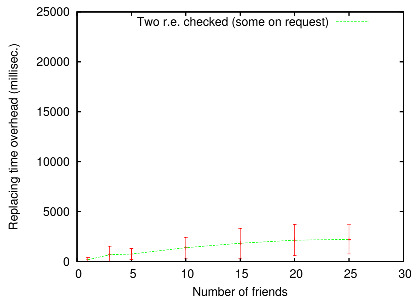
After investigating the influence of the number of regular expressions, we also investigated the influence of the specific browsing operations that we observed being more time demanding—the ones that involve Ajax requests. Hence, we designed a more time efficient solution for such operations—at a price of a reduced transparency and reduced functionalities for these operations. In the alternative implementation,



(a) All implemented regular expressions (r.e.) active.



(b) Two regular expressions (r.e.) active.



(c) Two regular expressions (r.e.) active. Some replacements made on request.

Figure 5: Replacing time overhead of FaceVPSN (average values, and standard deviation in bars).

we require the user to be aware of FaceVPSN for these operations. For example, when the user searches someone's profile in the search bar, we require the user to write the complete real name of the friend and then press `F8`. In this way, we remove the functionality of finding matching names while the user writes, but still the user is able to use the search bar and find the friend with his real name. Even if the results obtained in the previous experiments might be already satisfying, we argue that if the user is willing to sacrifice such described type of features, FaceVPSN can achieve even better performances. We run again the previous experiments of the setting with two regular expression, requiring an active user's participation for the replacements in browsing steps 8, 13, and 16 (Figure 4). The results are shown in Figure 5(c). From this figure, we can observe that the performances are significantly further improved when compared to the case of fully transparent behaviour (i.e. Figure 5(b)). Considering again the case of 25 friends in the user's VPSN, the replacement overhead decreases from 5,145 milliseconds to 2,220 milliseconds, on average. The standard deviation of this overhead also decreases, as described by the bars in the graphs.

Based on our experience with FaceVPSN, when a user navigates to some page in Facebook, the first replacements can be seen in less than one second (on average) after the new page is shown. This varies according to what the user is viewing. In case of a profile page with only a few wall posts, then the replacements are performed in a few seconds. Conversely, if the user is viewing the home page of Facebook, and there happens to be a lot of news feeds with friends, whose information needs to be replaced, then it takes more time.

6. CONCLUSION

In this paper we introduced the concept of VPSNs—Virtual Private Social Networks (applied to the social networks), that is similar to the concept of Virtual Private Networks (applied to computer networks). VPSNs allow the building of private social networks, the information of which is not available outside, leveraging the infrastructure already present for Social Networks Sites (SNSs). We implemented our concept through FaceVPSN, a VPSNs system for Facebook, one of the most widely used SNSs. FaceVPSN is a completely distributed system that implements VPSNs and mitigates the privacy problem in Facebook. To the best of our knowledge FaceVPSN is the first privacy solution for Facebook that is all at once: i) completely distributed; ii) Facebook independent—cannot be removed by Facebook; iii) hides public information from users outside the VPSN, from Facebook, as well as from Facebook applications. The implementation shows the feasibility of the proposal, while experimental results confirm its efficiency and usability.

Future research aims to optimize FaceVPSN to further reduce the overhead and to make it more user friendly. That is, to make the way XML files are shared more transparent, e.g. transparently leveraging the messaging system of Facebook itself. To further protect the real information stored on the local machine (to which more than one user might have access), FaceVPSN could also store the real information in an encrypted way and decrypt them when needed, based on a per user access. Finally, the automatic evolution of the views, as defined in Section 4, is also subject to investigation. For example, a view that a user has of someone's profile might automatically evolve according to some trust rules, or depending on the type of relations between various VPSNs' members.

7. ACKNOWLEDGMENTS

This work is partly funded by the European Network of Excellence NESSoS contract no. FP7-256980.

8. REFERENCES

[1] Virtual private social networks website. `http://sites.google.com/site/fbprivacy2010/`.

[2] Please rob me, June 2010. `http://pleaserobme.com/`.

[3] E. Aimeur, S. Gambs, and A. Ho. Upp: User privacy policy for social networking sites. In *Proceedings of the Fourth International Conference on Internet and Web Applications and Services*, pages 267–272. IEEE Computer Society, 2009.

[4] E. Aimeur, S. Gambs, and A. Ho. Towards a privacy-enhanced social networking site. In *International Conference on Availability Reliability and Security*, volume 0, pages 172–179. IEEE Computer Society, 2010.
[5] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin. Persona: an online social network with user-defined privacy. In *Proceedings of the ACM Conference on Data Communication*, pages 135–146. ACM Press, 2009.
[6] D. M. Boyd and N. B. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication.*, 13(1):Article 11, 2007.
[7] L. A. Cutillo, R. Molva, and T. Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *IEEE Communicaions Magazine*, 47(12), December 2009.
[8] B. Dybwad. Facebook and others caught sending user data to advertisers, 2010. `http://mashable.com/2010/05/20/facebook-caught-sending-user-data-to-advertisers/`.
[9] Facebook. `http://www.facebook.com`.
[10] Facebook. Facebook's privacy policy. `http://www.facebook.com/policy.php`.
[11] A. Felt and D. Evans. Privacy protection for social networking apis. In *W2SP '08: in Conjunction with the 2008 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2008.
[12] R. Gross and A. Acquisti. Information revelation and privacy in online social networks. In *Proceedings of the 2005 ACM workshop on Privacy in the Electronic Society*, pages 71–80. ACM Press, 2005.
[13] S. Guha, K. Tang, and P. Francis. Noyb: Privacy in online social networks. In *Proceedings of the First Workshop on Online Social Networks*, pages 49–54. ACM Press, 2008.
[14] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava. Anonymizing social networks. Technical Report 07-19, University of Massachusetts Amherst, March 2007.
[15] M. Kacimi, S. Ortolani, and B. Crispo. Anonymous opinion exchange over untrusted social networks. In *SNS '09: Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, pages 26–32. ACM Press, 2009.
[16] A. Korolova, R. Motwani, S. U. Nabar, and Y. Xu. Link privacy in social networks. In *Proceeding of the 17th ACM conference on Information and Knowledge Management*, pages 289–298. ACM Press, 2008.
[17] M. M. Lucas and N. Borisov. Flybynight: Mitigating the privacy risks of social networking. In *Proceedings of the 7th ACM workshop on Privacy in the Electronic Society*, pages 1–8. ACM Press, 2008.
[18] W. Luo, Q. Xie, and U. Hengartner. Facecloak: An architecture for user privacy on social networking sites. In *Proceedings of the 2009 International Conference on Computational Science and Engineering*, pages 26–33. IEEE Computer Society, 2009.
[19] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel. You are who you know: Inferring user profiles in online social networks. In *Proceedings of the third ACM International Conference on Web Search and Data Mining*, pages 251–260. ACM Press, 2010.
[20] Mozilla. Observer notifications. `https://developer.mozilla.org/en/Observer_Notifications`.
[21] Mozilla. Venkman javascript debugger project page. `http://www.mozilla.org/projects/venkman/`.
[22] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Proceedings of the 30th IEEE Symposium on Security and Privacy*, pages 173–187. IEEE Computer Society, 2009.
[23] A. of Exploits. Facebook's servers was hacked again by inj3ct0r team. `http://inj3ct0r.com/exploits/13403`.
[24] A. Sorniotti and R. Molva. Secret interest groups (sigs) in social networks with an implementation on facebook. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, pages 621–628. ACM Press, 2010.
[25] L. A. Sweeney. `http://groups.csail.mit.edu/mac/classes/6.805/articles/privacy/sweeney-thesis-draft.pdf`.
[26] N. Tabakoff. Facebook users are sitting ducks for identity theft, 2009. `http://www.dailytelegraph.com.au/news/facebook-users-sitting-ducks-for-identity-theft/story-e6freuy9-122580713389/`.
[27] L.-H. Vu, K. Aberer, S. Buchegger, and A. , Datta. Enabling secure secret sharing in distributed online social networks. In *Proceedings of the Annual Computer Security Applications Conference*, pages 419–428, 2009.
[28] W. Wolfe-Wylie. The harm of facebook pictures, 2010. `http://www.torontosun.com/life/2010/08/10/14978476.html`.
[29] A. L. Young and A. Quan-Haase. Information revelation and internet privacy concerns on social network sites: a case study of facebook. In *Proceedings of the Fourth International Conference on Communities and Technologies*, pages 265–274. ACM Press, 2009.
[30] E. Zheleva and L. Getoor. To join or not to join: The illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th International Conference on World Wide Web*, pages 531–540. ACM Press, 2009.

APPENDIX

A. IMPLEMENTATION SETTINGS

This section gives further details on the implementation and the experimental settings. Table 2 summarizes the regular expressions we defined in order to match a friend name in several locations. Note that links to be matched might appear for the same scenario either with "?" or "&". Hence, we considered both cases in the defined regular expression. For example, regular expressions 3 and 4 consider these two possible alternatives while searching for a username in the search list. Also, note that "\\\\" is used to match the character "&". Table 3 summarizes a set of regular expressions that are used in combination with User ID or Username. Finally, Table 4 describes the browsing sequence path used in the experiments.

Number	Regular Expression	Matches links with
1	`afriend.username`	username (e.g. name.surname)
2	`afriend.id`	id (e.g. 11111111)
3	`afriend.username+"\\\?ref=search$"`	username in search list ("?" case)
4	`afriend.username+"&ref=search$"`	username in search list ("&" case)
5	`afriend.id+"\\\?ref=search$"`	id in search list ("?" case)
6	`afriend.id+"&ref=search$"`	id in search list ("&" case)
7	`afriend.username+"\\\?ref=sgm$"`	username in pop-up windows ("?" case)
8	`afriend.username+"&ref=sgm$"`	username in pop-up windows ("&" case)
9	`afriend.id+"\\\?ref=sgm$"`	id in pop-up windows ("?" case)
10	`afriend.id+"&ref=sgm$"`	id in pop-up windows ("&" case)
11	`afriend.username+"\\\?ref=ts$"`	username in suggestions list ("?" case)
12	`afriend.username+"&ref=ts$"`	username in suggestions list ("&" case)
13	`afriend.id+"\\\?ref=ts$"`	id in suggestions list ("?" case)
14	`afriend.id+"&ref=ts$"`	id in suggestions list ("&" case)

Table 2: List of Regular Expressions to match the name in several places.

Number	Regular Expression	Matches with
1	`buddy_list`	same word to find users in chat list
2	`Chat with`	same word that appears in user's profile (it is used to replace user's first name in Chat with link in profile page)
3	`messages`	Message to link in profile page
4	`MessageComposer`	Send Friend a Message link in profile page
5	`video`	View Videos of Friend link in profile page
6	`can_poke`	Poke Friend link in profile page
7	`friend_suggester_dialog`	Suggest Friends to Friend link in profile page
8	`notes`	Subscribe to Friend's Notes link
9	`share_posts`	Subscribe to Friend's Links link
10	`mutual$`	"Mutual photos" link in photos profile section
11	`profile.php\\\?id="+afriend.id+"$`	Friend's Profile Link while in photos section
12	`photo_search`	Photos of Friend link while in photos section
13	`photos&viewas`	View Photos of Friend link in profile page
14	`ref=mf`	Friend's Name link when tagged in albums or videos
15	`photo_comments.php`	Friend's Photo Comments link
16	`photo`	Back to Friend's Photo link

Table 3: List of Regular Expressions used in combination with User ID or Username

Browsing Step	Browsing Operation	Automatic / On Request
1	Browsing home page	Always automatic
2	Browsing profile page	Always automatic
3	Browsing all comments in a wall post	Always automatic
4	Browsing info tab in profile page	Always automatic
5	Browsing photos tab	Always automatic
6	Browsing one picture	Always automatic
7	Browsing another picture	Always automatic
8	Replace name in search bar	Automatic or on request (pressing `F8`)
9	Browsing searched friend's profile page	Always automatic
10	Browsing home page	Always automatic
11	Browsing opened chat mini-window of friend's message	Always automatic
12	Browsing profile of friend who sent post	Always automatic
13	Replace name in search bar	Automatic or on request (pressing `F8`)
14	Browsing searched friend's profile	Always automatic
15	Browsing an event page which some friends attended	Always automatic
16	Browsing pop-up of attendands	Automatic or on request (pressing `F9`)
17	Browsing home page	Always automatic
18	Browsing message page	Always automatic
19	Browsing event page	Always automatic
20	Browsing photo page	Always automatic

Table 4: Test Browsing Sequence Path.