Scramble! your social network data

Filipe Beato[1], Markulf Kohlweiss[1,2], and Karel Wouters[1]

[1] Katholieke Universiteit Leuven
Dept. Electrical Engineering - ESAT/SCD/IBBT-COSIC
Kasteelpark Arenberg 10, Leuven-Heverlee (Belgium)
[2] Microsoft Research, Cambridge, UK

Abstract. Social network sites (SNS) allow users to share information with friends, family, and other contacts. However, current SNS sites such as Facebook or Twitter assume that users trust SNS providers with the access control of their data. In this paper we propose Scramble, the implementation of a SNS-independent Firefox extension that allows users to enforce access control over their data. Scramble lets users define access control lists (ACL) of authorised users for each piece of data, based on their preferences. The definition of ACL is facilitated through the possibility of dynamically defining contact groups. In turn, the confidentiality and integrity of one data item is enforced using cryptographic techniques. When accessing a SNS that contains data encrypted using Scramble, the plugin transparently decrypts and checks integrity of the encrypted content.

1 Introduction

Social Network Sites (SNS) such as Facebook, MySpace, LinkedIn, and Twitter- are becoming increasingly popular. Millions of users access these sites as part of their daily routine. These sites provide technological features that allow users to share content and build communities around shared interests. Users can assess, analyse, and modify privacy preferences made available by the service providers, but they cannot control the enforcement of these preferences.

SNS users often post a large amount of privacy sensitive information on SNS, such as their date of birth, their daily activities, or political views. As already mentioned, users have to rely on privacy preferences enforced by the SNS providers to protect this data. However, these policies and privacy preferences are often extremely coarse and difficult to locate [19], which lead to potential misconfigurations [7]. Nevertheless, the SNS provider still has access to all users' data and can share it with external parties, like targeted advertisement companies. Therefore, the user does not have full control over his data. In addition, SNS may offer application programming interfaces that may expose and share the users' information with other services. Finally, policies may be changed intentionally by providers, to help them strike a balance between the interests of advertisers, application providers, and usability.

All of this may leads to serious privacy concerns. The need for a mechanism that returns control over both access-control policy configuration and enforcement for user-generated content to the users themselves has been identified in previous works [1, 2, 12, 14, 15, 18]. Clearly, this is highly desirable for SNS, but is also relevant for other Web 2.0 services.

In this paper, we present Scramble, a client side application implemented as a Firefox extension to help users keep their data confidential. Scramble allows users to encrypt their posted content in the SNS. Therefore, Scramble guarantees confidentiality of users' data towards the SNS-provider. To support audience segregation [11, 20], scramble contains an easy-to-use user interface for defining the set of users the user's content should be shared with.

Our implementation of Scramble is SNS independent and is suitable for immediate deployment as open source software. We make use of the OpenPGP[3] standard to enforce confidentiality and integrity. Several SNS providers have a length limitation for posted content, e.g., Twitter[4], and do not allow publication of encrypted text defined on their Terms of Service, like Facebook. For those reasons, we provide an implementation of an external tiny link server. The server stores the encrypted data and produces a short link that works as an index to the posted encrypted content. The scramble prototype is part of research performed within the EU-PrimeLife[5] project.

The remainder of this paper is organised as follows: In Section 2 we review related work and compare it with our proposal. We introduce our goals and assumptions in Section 3. In Section 4 we present a detailed description of the Scramble design, and in Section 5 we describe our implementation. Section 6 gives a security, performance and usability analysis of our implementation. Finally, in Section 7 we discuss future work and conclude by summarising our results.

2 Related Work

We discuss existing approaches for enforcing access control rules in SNS: Social network providers, such as Facebook and MySpace, implement access control mechanisms for user-generated data. These mechanisms, however, offer no protection against the SNS providers themselves, since they through their control of the servers running the service have access to all of a user's information. To avoid access by the SNS, Lockr [3] hides pictures posted in the SNS by replacing it by a link, and storing the picture at a third party server in unencrypted format. This approach relies on a third-party that might not be trustworthy – instead of trusting the SNS provider one now has to trust the third party server. In [8] the authors apply the concept of virtual private networks to social networks. Whilst, this solution is SNS independent and allows users to replace the

[3] OpenPGP represents the IETF RFC 4880 - `http://www.openpgp.org/`

[4] Twitter allows a maximum of 140 characters per post

[5] This project aims at providing significant improvements to protect privacy in emerging digital world. `http://www.primelife.eu`

original attribute data with some pseudo information. The real information is then sent and stored in friends machines. Thus, besides creating a bargain on friends machines instead of delegating to the server, it does not allow users to selective enforce access control over their posted data.

There are several proposals that use encryption to protect a user's information that target Facebook. flyByNight [16] is a Facebook application that protects user data by storing it in encrypted form in Facebook. This application is Facebook dependent and relies on Facebook servers for its key management. The decryption algorithm is implemented in JavaScript and is retrieved from the Facebook application. Thus, while browser independent, it is not secure against active attacks by the provider – Facebook. In contrast, Scramble is a client side application that has no SNS dependencies.

NOYB (*None Of Your Business*) [13] is a system that targets Facebook and uses encryption to protect private information. The personal details of users, such as name and gender, are divided into multiple pieces of data, called atoms. These atoms are separated and shuffled with atoms of other users, acting as a random substitution cipher. The encryption method used by NOYB just replaces the privacy details of user A with those of random users B and C. Only the user himself and his friends can reverse the process and reconstruct the profile. However, this can only be applied to the personal details on the user's profile, and does not allow encryption of free text entries as frequently found in social networks.

FaceCloak [17] is a Firefox extension that uses a symmetric key to encrypt user's information in Facebook. The encrypted data is stored in the FaceCloak server, and replaced in Facebook by random text fetched from wikipedia. The symmetric keys are shared with the set of users authorised to read the content. The random text acts as an index to the encrypted data on the server.

One of the problems with the FaceCloak and NOYB model is, that using random meaningful text retrieved from Wikipedia or other users may lead to social conflicts, if other users take them to be genuine user content.One could argue that the goal of natural-text as either cipher-text or index by NOYP and FaceCloak respectively is an important anti-censorship mechanism against a SNS that sees threads to it's advertising revenues. Should the need arise, Scramble could make use of similar techniques. However, we believe that other solutions to this dilemma, such as client-side privacy friendly advertising mechanisms may be more desirable.

Moreover, FaceCloak has a complicated and inefficient key distribution system. For each piece of content, the user accessing the content has to use an offline channel to retrieve the key. Scramble uses a simpler and more reliable approach for key distribution. The encryption of the content is done using public keys, and thus a user with access rights just needs to use his own secret key for decryption. As a usability compromise we restrict the use of PGP's web-of-trust mechanism to power-users and adopt leap-of-faith authentication as the default key-distribution paradigm.

The schemes defined above have proposed mechanism to protect users' sensitive information in Facebook. However, they are Facebook dependent, while Scramble is SNS independent.

Diaspora[6] presented a new privacy friendly, open source social network. The project offers users the possibility to share privately information using OpenPGP mechanisms, like Scramble. It uses its own distributed network for storing the encrypted data. However, while it offers a new service to protect the privacy of its users, it does not support the existing and highly popular centralised social networks services.

3 Goals and Assumptions

We represent a social network as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, whose vertices represent users and whose edges represent the undirected connections between users. Each u establishes a set of relationships $\mathcal{R}_u \in \mathcal{V}$ that contains all users to which u has a connection. Formally, $(u, v) \in \mathcal{E}$ if and only if $v \in \mathcal{R}_u$.

We now describe our thread model and our assumptions, as well as the goals of our system.

Threat model. Our threat model considers the SNS providers as potentially adversarial. SNS providers have access to all of the user's private information. SNS providers can leak information to external parties and have the power to tamper or replace user generated content on the SNS. Therefore, users may be vulnerable to data leakage, impersonations and false judgements.

We consider curious users seeking sensitive information to be a weaker adversary than the provider. Such users benefit from the SNS as a channel to listen and obtain sensitive content from other SNS users and may use it for their own profit. Providers commonly enforce default privacy settings to protect against such threat, but these settings are often permissive [2] and subject to frequent change. Thus, users lack control about which other SNS users can access their content.

We rely on the integrity of users' personal environment, such as their browser and computer. We assume that no external party has access to or can compromise a user's environment. We assume that each user u has a public and secret key pair (pk_u, sk_u), where pk_u is known by all $\mathcal{R}_u$ and sk_u is only known by u. We assume that users u and v exchange their public keys when a friendship relation is established using an authenticated offline channel.[7]

Goals. Users need to be able to control their own data, and specify who can access it, preferably without relying on third party servers, such as the SNS

[6] Diaspora: `https://joindiaspora.com/`

[7] For the sake of reducing the entrance barrier for ordinary users, we will, sometimes willingly break the last assumption and allow users to start communicating using unauthenticated keys. Users are, however, advised to check the authenticity of keys using key fingerprints, and to get suspicious if keys change without premonition.

providers. Any user u can create new content d for the SNS, e.g., as a wall post or some other message. Thus, the desired goals for Scramble are the following:

Privacy Preservation: A user u should be able to define the subset $\mathcal{S}_d$ of recipients from $\mathcal{R}_u$ that are authorised to read d. Only users in $\mathcal{S}_d$ are able to read d. Both $\mathcal{S}_d$ and the content of d should be kept hidden from the provider. The confidentiality of d should be protected by cryptographic techniques. However, once d is distributed among the users in $\mathcal{S}_d$, there is no way to prevent a *malicious* user in $\mathcal{S}_d$ from storing or re-distribute the content of d. In this case, the receiving user is said to break the social contract associated with the establishment of the friendship relation.

Publisher Integrity: Scramble should guarantee d's integrity when posting d in the SNS using cryptographic techniques. This prevents attackers from tampering with the content of d and impersonating u.

Deployability: Scramble is meant to be deployed in the real world. Thus, it must be stable, compatible with different environments, and SNS independent.

Usability: Scramble should present a user interface that is easy to use. In order to overcome usability issues, such as those presented in [21], the operations should be simple and the cryptographic techniques transparent. Operations like the generation, import and export of keys should be effortless or hidden. If a user v is not authorised to read d, then Scramble should hide d from v.

4 Scramble

In this section we describe and motivate the design details and functionalities of Scramble. We first discuss design decisions specific to key management, access control policies, and the employed cryptographic mechanisms. Then, we describe the process flow of Scramble from a user perspective.

4.1 Key Management, Access Control Policies, and Cryptography

Key Management. In Scramble, each user u holds a OpenPGP key pair, composed of public key pair pk_u and a secret key pair sk_u. The public and private key pairs consist of the public respectively private keys of an ElGamal encryption and a DSA signature scheme. The keys can be either generated (default behavior) or imported (power-user behavior) by the user upon Scramble initialisation. If the user Alice[8] wants to share d with the set $\mathcal{S}_d$, she must possess the associated public keys pk of all users in $\mathcal{S}_d$. All pk of $\mathcal{S}_d$ are stored in Alice's machine, and are managed by Scramble.

Key management is a hard problem due to the possibility of key tampering and the fact that it is counterintuitive to ordinary users. A malicious user v or the SNS provider can replace the pk_u of the user u to impersonating u. Thus, it is

[8] For the sake of concreteness, we sometimes use Alice and Bob for the user u that posts a new d and the intended reader v respectively.

important that users can correctly distribute their public keys, as they are used for encryption when posting content. If users, however, are not able to exchange any keys and resort to unencrypted alternatives, they are even worse off.

Users have to be able to exchange their pk when a friendship connection is established. They can make their public key available using the provider or a key server and should verify fingerprints using an offline channel to verify the authenticity of a public key. As Scramble makes use of the OpenPGP standard we can make use of any public PGP server. We opted to verify the authenticity of keys manually as the current OpenPGP web of trust has proved to be too complicated for ordinary users [21]. Users have to either take the leap-of-faith or check the fingerprints. For future versions it should be easy to introduce a web-of-trust mechanism, if this is desired by power-users.

Alternatively, our key management model could be extended by making public keys available over an SNS-based mechanism such as the one proposed by [5], where users cross certify their digital certificates using SNS relationship connections. The cross certification is achieved by users signing other users' digital certificates, which are composed by the public key together with some Personal Identifiable Information (PII).

For key revocation or key update users are required to distribute a new public key. However, this only affects new content.

Access Control Policies. We consider that $\mathcal{R}_u$ is represented in Scramble by the public keys of the users in $\mathcal{R}_u$. Moreover, a user u can define groups $G_i \subset \mathcal{R}_u$ in order to separate $\mathcal{R}_u$ into categories.

Whenever u publishes a new document d in the SNS he can define with whom to share. For that, u selects a subset $\mathcal{S}_d$ from his $\mathcal{R}_u$ that is to be authorised to read d. $\mathcal{S}_d$ can be composed of single users $v_i \in \mathcal{R}_u$, of a set of pre-defined groups G_i or of a mix of both. The set $\mathcal{S}_d$ can be different for each d posted. For any $\mathcal{S}_d$ update d is required to be re-posted.

Figure 1 represents an example of our approach for defining access rights. Alice has relationships $\mathcal{R}_{Alice}$ and posts contents $\{d_i\}$. $\mathcal{R}_{Alice}$ is represented by three groups $Work, Friends, Family$ and a single relationship Bob. This helps Alice to define her $\mathcal{S}_d$ in an easier way. When Alice posts new content d she may, e.g., defines $\mathcal{S}_d = \{Bob \cup Work\}$. In this way, Alice keeps d private to a limited audience defined by $\mathcal{S}_d$. Moreover, the audience defined by $\mathcal{S}_d$ is only known to her.

Cryptographic primitives. For the confidentiality and integrity of d we had the choice between traditional hybrid-encryption techniques, like OpenPGP [22], or broadcast encryption such as [4, 6]. In both cases, the users' public keys in $\mathcal{S}_d$ would be used to create the access list that would be attached to the final posted content. The confidentiality of d is then achieved using an encryption algorithm, while integrity of d is assured by signing d before encryption.

$$d' \leftarrow \texttt{Encrypt}_{\mathcal{S}_d}(\texttt{Sign}(d, sk_u))$$

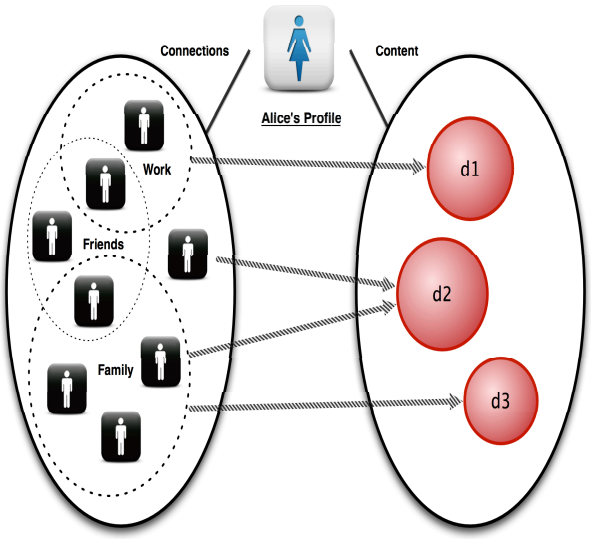

Fig. 1. Access control mapping example

We chose OpenPGP as it is a well deployed standard with support for multiple recipients encryption using hybrid encryption. Moreover, most broadcast encryption schemes such as [6] do not provide key privacy, with the exception of [4]. The latter, however, also uses a hybrid-encryption approach internally and does not offer performance advantages. We discuss weaknesses of OpenPGP that we are aware of in Section 6, but we believe that it is more reasonable to fix OpenPGP, than to abandon it as a design choice.

Thus, d is encrypted with a one time random-generated secret key k using a symmetric algorithm. Then, $|\mathcal{S}_d|$ encryptions of k are generated using the public key of each subject in $|\mathcal{S}_d|$. The integrity of d is assured by signing d before encryption. Hence, d is published as follows.

$$
\begin{aligned}
&\textit{Let } \mathcal{S}_d = \{Alice, Bob, Charlie\} \textit{ be set by } u \\
&\sigma_d \leftarrow \texttt{Sign}(d, sk_u) \\
&\texttt{C} \leftarrow \texttt{SymEnc}_k(\sigma_d || d) \\
&d' \leftarrow \{\texttt{PKEnc}_{pk_{Alice}}(k) || \texttt{PKEnc}_{pk_{Bob}}(k) || \texttt{PKEnc}_{pk_{Charlie}}(k) || \texttt{C}\}
\end{aligned}
$$

The public key encrypted values of k are appended to the symmetric encryption and represent an anonymous version of $\mathcal{S}_d$ that specifies which other users are allowed to see d. This will indeed increase the storage overhead on the server side, but it will save the user from managing a large number of different keys for every new d on his machine. In addition, this allows the user to keep his defined access sets anonymised, and enforce different access control rights for each document d. It is important to note, that OpenPGP uses a separate ElGamal encryption key and DSA signing key to perform the previous operations.

In order to keep the set of recipients hidden, we use the *hidden-recipient* option. This option conceals the key IDs of recipients in the encrypted content. In this way, only the users in $\mathcal{S}_d$ are able to retrieve the value of d. Other users, and the SNS provider stay oblivious of the raw value of d, learning only d'. However, the length of the output is directly affected by the size of $\mathcal{S}_d$.

4.2 User Interaction Flow

The Scramble system consists of two modules. The first and main element, Scramble, is a Firefox extension that contains the cryptographic primitives to enforce the access rights, and the key and group management. The second and optional element is a TinyLink server. This server just receives content posts and returns a link to the location of the content. We assume that users can choose their external server or set their own server with our provided implementation.

We describe the two elements using the flow of operations needed to to publish and retrieve data on a SNS. The process flow is preceded by an initialisation phase.

Initialisation. In this phase, Alice generates her key pair (pk_u, sk_u), uploads it to the key server, obtains keys for her contacts $\mathcal{R}_u$, and creates her groups G_i. In order to import her relationship contacts, Alice could, in future version, extract the contacts from the SNS provider directly using the mechanism described in [9]. For now, imports need to be done manually based on the email address of users.

Posting content. Alice is a user that wishes to post a new d in the SNS (Figure 2). Therefore Alice (1) selects $\mathcal{S}_d = \{Bob, Charlie, ..., Dave\}$ using Scramble. Then, Scramble signs d and encrypts d with the keys of the authorised users in $\mathcal{S}_u$. If the SNS limits the length of the posted d, then (2) Scramble posts d', the encryption of d, in the TinyLink server that returns a tiny link to the stored location. (3) Scramble posts the encrypted value d' or the tiny link to d' in the SNS. The value of d' is transmitted from Scramble in encrypted format, keeping a possible attacker oblivious.

Retrieving content. The decryption of encrypted content from the SNS is transparent to the user (Figure 3). First, (1) Scramble reads the encrypted value of d from the SNS. If the content is a tiny link, then (2) Scramble uses the tiny link retrieves d', the encrypted value of d from the TinyLink server. Subsequently, (3) Scramble tries to decrypt and if successful, verifies if d' was not tampered with and that it was in fact Alice who signed d. Since d came from Alice and Bob is in $\mathcal{S}_d$, Bob is authorised to read d. Thus, Scramble presents the value of d to Bob. Otherwise, the decryption fails, and the retrieved value d' is not shown.

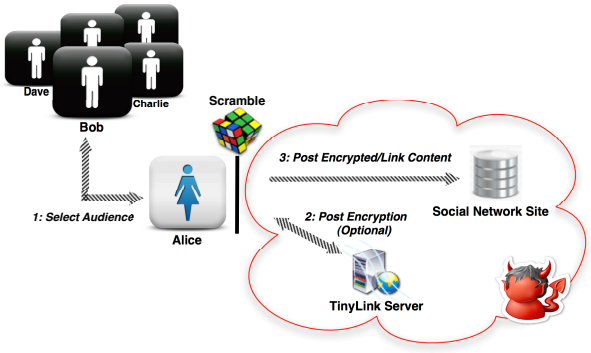

Fig. 2. Posting new Content Process

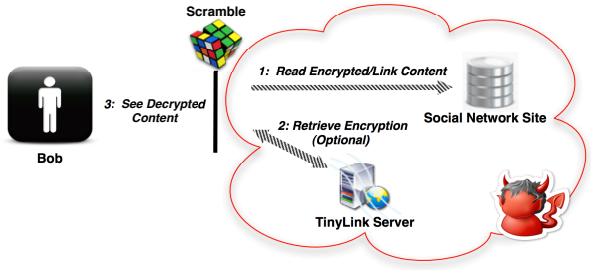

Fig. 3. Reading Content Process

5 Implementation

Our implementation represents the design functionalities in software. We have implemented Scramble as an open source application[9] under the EPL licence [10]. In this section we outline our implementation by describing the details of the application modules along with the functional aspects.

The main module of the implementation is the Firefox extension, that manages and enforces the access control lists. Our implementation is composed by the following two modules.

Firefox extension. Scramble is a client-side application implemented as a Firefox extension, that allows cross-platform client-side encryption and key management. Due to the fact that a Firefox extension is developed mainly in JavaScript,

[9] Scramble version can be found in the project website `http://tinyurl.com/ScrambleIt`.

we have used a Java XPCOM[10] component to improve performance of the cryptographic module. The Java XPCOM component contains an implementation of the OpenPGP standard. The component executes either a BouncyCastle[11] (BC) OpenPGP implementation or the GnuPG[12] binary module that implements the OpenPGP standard. By means of having the two different implementations, the user can choose to have an embedded OpenPGP implementation with a dedicated key ring with BC, or to execute the general GnuPG module with a key ring that can be shared with other programs.

Key management. The key management is handled by Scramble. The OpenPGP key pair can be generated or imported by the user Alice during installation, however, it can be changed afterwards. Alice can then upload her key to the public key server in order to allow her friends to download it. The group management and their definition is defined by Alice, by operating on a simple user-interface (Figure 4). The OpenPGP pk of all users $v \in \mathcal{R}_u$ are stored in Alice's machine, and act as their identification. In order to import those keys, Alice uses the key server mechanism to import it by referring to them using her friends email addresses. The fingerprint of an imported key pk_v of Bob can then used by Alice to verify the authenticity of his key. The full or a part of the fingerprint should be communicated using a secure offline channel. Thus, $\mathcal{R}_u$ in Scramble represents a subset of the relationship graph on the SNS.

Publish data operation. To perform the operation where Alice wants to post a new d into the SNS in encrypted format, Alice is required to write d into a field in the SNS and select it. Scramble then allows Alice to define the $\mathcal{S}_d$ for d from her circle of trust $\mathcal{R}_u$. The values of the pk of each users in $\mathcal{S}_d$ are loaded and used in the encryption algorithm. Scramble retrieves the selected text from the DOM[13] tree of the SNS website's displayed HTML page and posts the value of d in encrypted format or the tiny link to d', the encrypted value of d. This is only readable by users in $\mathcal{S}_d$.

Retrieve data operation. For the decryption operation, Scramble parses the DOM tree of the website and searches for Scramble tags representing encrypted blocks of text. If the user belongs to the set and thus has access, Scramble automatically and transparently decrypts the content presenting the unencrypted data d to the user. Otherwise, the data is hidden or is indicated by a pre-defined message, like "Non-authorised content". This is done by only replacing the encrypted text independently from the DOM's tree style.

In order to perform the operations to store and retrieve data from the tiny link server, Scramble executes `XMLHttpRequest` `Post` and `Get` calls in JavaScript.

[10] `http://www.mozilla.org/projects/xpcom/`
[11] `http://www.bouncycastle.org/`
[12] `http://www.gnupg.org/`
[13] `http://www.w3.org/DOM/`

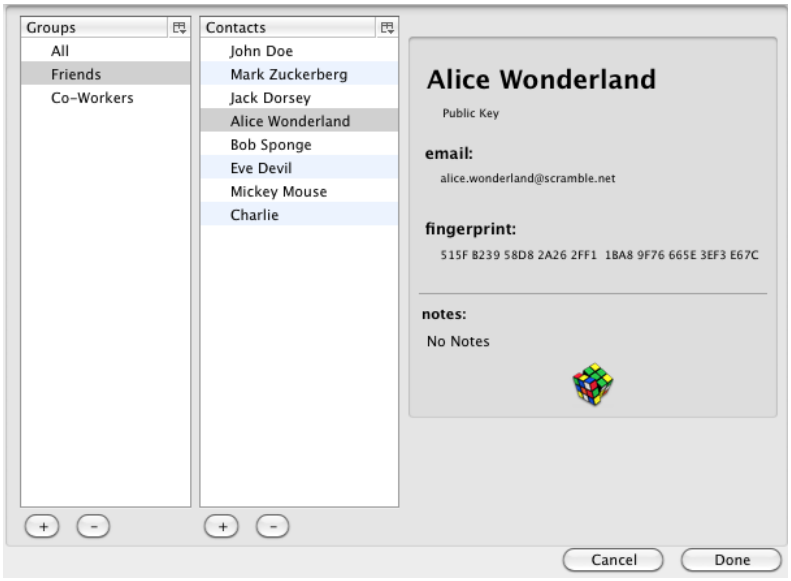

Fig. 4. Access Control Definition User Interface

Users that are not using Scramble will see the encrypted data (Figure 5). This can be either the full encrypted block d' or a link to the block.

Tiny Link Server. The Tiny Link Server was developed to target the limitation of content size imposed by SNS providers. The PHP[14] server stores encrypted data and returns a tiny link (short URL) which represents the index of d', the encryption of d. This server can be controlled by the users directly or outsourced into a cloud server, that may or may not require extra authentication. We provide the users with the source code and details for their own implementation.

6 Security Analysis, Performance & Usability

In this section we proceed with a security analysis of our implementation. Then, we present our performance and usability results.

Security analysis. We analyze the resilience of the current implementation of scramble against a number of potential attacks.

Recipients Set Anonymity Scramble keeps the content document d confidential using OpenPGP encryption. In order to anonymise the recipients set for outsiders, Scramble uses the OpenPGP option *hidden-recipients* to conceal the key

[14] `http://php.net/`

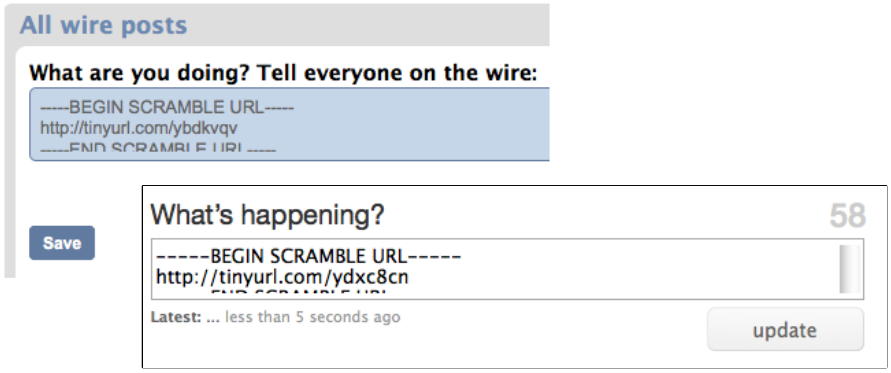

Fig. 5. Scramble in *(Private)* Twitter

IDs. However, this does not offer anonymity of the set of recipients towards a malicious user in the set, as shown in [4]. We note that Scramble does not provide protection against traffic analysis, meaning that the provider could infer who has access to the content by analysing download and upload operations. Protection against this kind of attacks is left as a subject of future work.

Active Attacks In an active attack, a malicious service provider attempts to tamper with the content item, by compromising content integrity and confidentiality. In Scramble the user posts the item d in encrypted format on the server to ensure the confidentiality of d. A malicious server can also have the objective of fooling or impersonating users by changing or replacing d. In order to prevent such attacks, the user posts d together with a signature on d in encrypted format.

Performance. To be usable, Scramble must minimise its implementation overhead. The use of an XPCOM component allows to execute either a BC Java OpenPGP implementation or the binary command line GnuPG module. Both of which provide very efficient encryption, decryption and signing operations.

In order to analyse the performance of our implementation, we focus on the cryptographic algorithms that represent the most expensive operations and the response of the tiny link server, which include the network latency and server process. Therefore, Scramble depends directly on the amount of recipients $r = |\mathcal{S}_d|$ per encrypted block d. The encryption and decryption costs are represented in Figure 6, where the size of the contact set for encryption and decryption operations goes from 0 to 720 contacts[15]. The public key operations are the most costly operations compared to the use of a symmetric encryption algorithm. The performance complexity details are described as follow.

[15] Tests performed on a 2GHz AMD Athlon(tm) XP 2400+, with 1Gb RAM

Publish operation. is affected by the efficiency of the encryption and signing algorithm E and t_s. The efficiency of E is directly affected by r. Thus, the overall performance is $t_s + O(r)$.

Retrieval operation. is affected by the number of encrypted items per page n, by the efficiency of decryption and verification algorithm D and t_s. Whilst a user in $\mathcal{S}$ is required to perform an average $r/2$ decryptions, a user that does not have access rights is required to perform r. Thus, the overall performance for retrieving information is $n(t_s + O(r))$.

Usability. We have performed some user tests with local Belgian students, where Scramble was well received in terms of user experience and functionality. Scramble was also submitted to a usability expert evaluation conducted by KAU[16] in the scope of the EU-PrimeLife project. However, a more advanced user experience test targeting a larger audiences is left for future work.

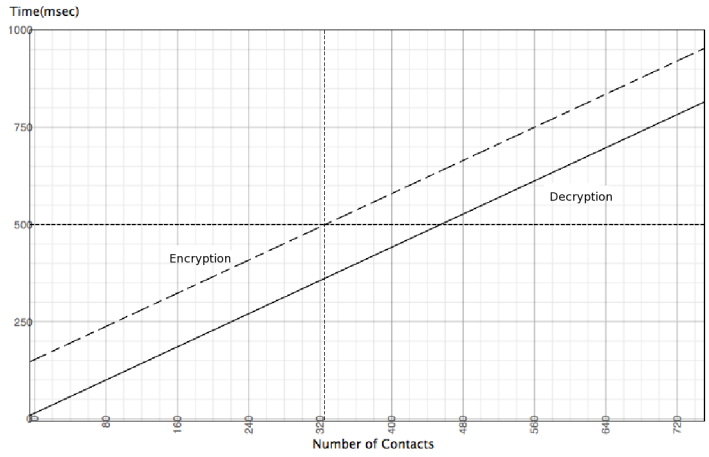

Fig. 6. Performance of Scramble operations per contact set

7 Future Work & Conclusions

In this section we start to enumerate some discussion points on the current implementation and future directions. Then, we conclude by presenting our results.

[16] Karlstad University - `http://www.kau.se/`

Future work. At the moment, it is the user himself who is responsible for defining $G \subset \mathcal{R}_u$. In the future, we intend to extend Scramble to be able to infer the privacy policies information from social network specific tags during an initialisation state. These tags are added to the content or can be derived from the context, as shown in [9].

In order to attract a large set of users and extend to other systems, we are currently developing a Scramble version as a Google Chrome application. In addition, a mobile device extension of Scramble would be attractive.

Conclusions. We designed and implemented Scramble, a Firefox extension that allows users to define and enforce selective access control preferences for data published on social network sites. Scramble is SNS independent and can be used in diverse SNS, like Twitter, Facebook, Clique[17] and MySpace. Through the integration into a Firefox extension, the encrypted content is automatically decrypted by the browser for authorised users. The extension also allows the definition of groups to define audience segregation, and the encryption of content under the keys of all group members.

Using a public key encryption scheme we are able to protect the integrity and confidentiality of user created data, especially towards the SNS provider, by means of encryption. At the moment, the implementation just allows content encryption as wall posts, private messages and news status. However, it is also possible to extend to other content types, such as pictures, by following the same directions.

Due to the fact that it has been designed to be general and SNS independent, it can also be used with other Web 2.0 services, such as blogs, forums and wikis. Potentially, it allows users to store data in encrypted format in any cloud service.

8 Acknowledgements

This work have been supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government, by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and in part by the European Commission through the ICT program under the following contract: ICT-216483 PRIMELIFE. The authors would like to thank Ronald Leenes and Jan Camenisch for the useful discussions, and Venelin Gornishki and Elmar Tischhauser for helping during the development, test and dissemination phases of the tool.

References

1. *(Under)mining Privacy in Social Networks*, 2008. Google Inc.
2. Alessandro Acquisti and Ralph Gross. *Imagined Communities: Awareness, Information Sharing, and Privacy on the Facebook*. 2006.

[17] `http://clique.primelife.eu/`

3. Geoff Salmon Amin Tootoonchian and Ahmad Ziad Hatahet. Fine grained access control in online social networks. Technical report, 2007.
4. Adam Barth, Dan Boneh, and Brent Waters. Privacy in encrypted content distribution using private broadcast encryption. In *In Financial Cryptography 06*, page 2006. Springer. LNCS, 2006.
5. Patrik Bichsel, Samuel Müller, Franz-Stefan Preiss, Dieter Sommer, and Mario Verdicchio. Security and trust through electronic social network-based interactions. *Computational Science and Engineering, IEEE International Conference on*, 4:1002–1007, 2009.
6. D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Crypto*, pages 258–275, 2005.
7. Joseph Bonneau and Sren Preibusch. The privacy jungle: On the market for data protection in social networks. In *The Eighth Workshop on the Economics of Information Security (WEIS 2009)*, 2009.
8. Mauro Conti, Arbnor Hasani, and Bruno Crispo. Virtual private social networks. In *Proceedings of the First ACM Conference on Data and Application Security and Privacy (ACM CODASPY 2011)*, page to appear, 2011.
9. George Danezis. Inferring privacy policies for social networking services. In *AISec '09: Proceedings of the 2nd ACM workshop on Security and artificial intelligence*, pages 5–10, New York, NY, USA, 2009. ACM.
10. Eclipse Foundation. Eclipse public license (epl) frequently asked questions, 2007. Accessed Dec. 2007.
11. Erving Goffman. *The Presentation of Self in Everyday Life*. Doubleday, Garden City, New York, 1959.
12. R. Gross and A. Acquisti. Information revelation and privacy in online social networks (the Facebook case). In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80.
13. Saikat Guha, Kevin Tang, and Paul Francis. Noyb: privacy in online social networks. In *WOSN '08: Proceedings of the first workshop on Online social networks*, pages 49–54, New York, NY, USA, 2008. ACM.
14. Ali Khajeh-Hosseini, Ian Sommerville, and Ilango Sriram. Research challenges for enterprise cloud computing. *CoRR*, abs/1001.3257, 2010.
15. Balachander Krishnamurthy and Craig E. Wills. Characterizing privacy in online social networks. In *WOSN '08: Proceedings of the first workshop on Online social networks*, pages 37–42, New York, NY, USA, 2008. ACM.
16. Matthew M. Lucas and Nikita Borisov. Flybynight: mitigating the privacy risks of social networking. In *Proceedings of the 7th ACM workshop on Privacy in the electronic society (WPES)*, pages 1–8, New York, NY, USA, 2008. ACM.
17. Wanying Luo, Qi Xie, and Urs Hengartner. FaceCloak: An architecture for user privacy on social networking sites. In *2009 International Conference on Computational Science and Engineering (CSE)*, volume 3, pages 26–33, Los Alamitos, CA, USA, August 2009. IEEE.
18. San-Tsai Sun and Konstantin Beznosov. Open problems in web 2.0 user content sharing. Jun 2009.
19. New York Times. Facebook privacy: A bewildering tangle of options. `http://www.nytimes.com/interactive/2010/05/12/business/facebook-privacy.html`.
20. Bibi van den Berg and Ronald Leenes. Audience segregation in social network sites. In *SocialCom/PASSAT*, pages 1111–1116, 2010.
21. Alma Whitten and J. D. Tygar. Why johnny can't encrypt: a usability evaluation of pgp 5.0. In *Proceedings of the 8th conference on USENIX Security Symposium - Volume 8*, pages 14–14, Berkeley, CA, USA, 1999. USENIX Association.

22. Philip R. Zimmermann. *The official PGP user's guide*. MIT Press, Cambridge, MA, USA, 1995.