PrivacyJudge: Effective Privacy Controls for Online Published Information

Bastian Könings, David Piendl, Florian Schaub, and Michael Weber
Institute of Media Informatics
Ulm University, Germany
{ bastian.koenings | david.piendl | florian.schaub | michael.weber }@uni-ulm.de

***Abstract*—With the rise of online social networks, sharing of personal information online has become the rule for many people rather than the exception. However, users have only limited abilities to effectively control privacy of their posted information. Even gaining an overview of posted information is already a difficult task. We propose a privacy control system for personal information online. PrivacyJudge allows fine-grained access control to posted information and provides a comprehensive overview of previously posted data. Thereby, PrivacyJudge does not require cooperation of service providers, instead operation remains transparent to them. Thus effectively providing privacy protection against other users and service providers alike. PrivacyJudge follows a hybrid approach combining cryptographic enforcement with social signaling to achieve privacy protection beyond technical enforcement boundaries by leveraging system trust as well as social trust mechanisms.**

I. Introduction

With the rise of online social networking sites more and more people are publishing more and more personal information online. Be it vacation photos, party pictures, personal blog posts or wall comments – people publish the information online and want to share it with their friends and associates. But once published online users lose autonomy over their data. Personal data is often under the control of service providers, such as online social networking sites, which offer only limited control over how other users or the general public may access the data. When it comes to internal data retention, users are at the good will of the service provider, which may comply with a deletion request of a data item or just hide it from view – the actual behavior is not clear to the user and outside her control. Other users may also copy the data and distribute it further. This kind of data dissemination is not under control of single service providers anymore. Furthermore, the Streisand effect suggests that trying to get some data deleted may only spurn its distribution [1].

Nevertheless, in recent years the issue of online privacy has been gaining focus with larger parts of the population, as evidenced by frequent coverage of privacy issues and concerns in mainstream media. Recent studies indicate that already teenagers and children are aware of privacy risks when posting data online [2]. But it is difficult for users to keep track on what information has been posted where and who has access to it. What is missing so far are technical means that provide users with an overview of published data and help them control who may access it. Such systems should not only protect personal privacy against other users but also against service providers which may have a questionable view on privacy, e.g., large social networking sites. Thus, such privacy control systems need to be transparent to the service provider in order to prevent blocking.

In this work we propose PrivacyJudge as an easy to use system for controlling who can access what posted information and for how long. In contrast to most related work, we acknowledge that it is nearly impossible to fully control the distribution of information. More importantly, in any system with human-computer interaction there is a certain point beyond which technical enforcement is not feasible anymore. Even when we assume a fully trusted computing system, in which the users can only access data in a trusted viewer which prevents storage or extraction of personal information, it cannot be prevented that human users manually write down the information, memorizes it, or simply take a picture of the screen. Therefore, PrivacyJudge follows a hybrid approach for online privacy control to address this issue. We combine a policy-based cryptographic enforcement system with social signaling techniques.

The rest of the paper is structured as follows. In Section II we discuss and analyze related work. Section III gives clear requirements definitions for an online privacy control system. Section IV outlines the PrivacyJudge system and describes the proof-of-concept implementation as a Firefox extension. Section VI gives a security analysis of the PrivacyJudge system and a comparative study with related systems. Section VII concludes the paper.

II. Related Work

Existing work on privacy-aware information publishing can be divided into three categories: Solutions for existing Online Social Networks (OSNs), new decentralized OSN solutions, and global approaches. In addition to these categories we will discuss optimistic approaches, which aim to enhance expected privacy by different signaling mechanisms.

A. Solutions for existing OSNs

NOYB (None Of Your Business) [3] obfuscates the information items of a user's profile (e.g., name, gender, age) by a pseudorandomly substituting them with items from other user profiles. The main drawback of this approach is that a profile contains random but real information from other users.

3rd IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT 2011), Boston, USA, October 2011
Author version. Definitive version at: http://dx.doi.org/10.1109/PASSAT/SocialCom.2011.86

FaceCloak [4] obfuscates published information by utilizing random generated text, e.g., the random article feature of Wikipedia. Original information is encrypted with a master key and stored on a third party server. For distribution of master keys the authors propose to use emails.

The Facebook application FlyByNight [5] provides encryption of user messages based on public-key encryption and proxy re-encryption. Similar to FaceCloak, encrypted messages are stored on an external server.

Persona [6] allows OSN users to apply access control policies to their information with a group-encryption scheme based on associated profile attributes, e.g., "neighbor" or "football fan". Sensitive information is stored encrypted on a storage service and is referenced by tags that are published through an application interface to the OSN. A user client, implemented as a browser plugin, handles the lookup of resources and performs cryptographic operations.

B. Decentralized OSN solutions

A general approach to avoid that a single service provider is in full possession of user data, is the deployment of completely decentralized architectures.

PeerSoN [7] is a decentralized peer-to-peer OSN. It utilizes a DHT for data and user lookups and performs data exchange between peers, who are the OSN users themselves. User profiles are randomly distributed in the network. The integration of encryption mechanisms is left to future work and not included in their current prototype.

A similar peer-to-peer OSN is Safebook [8]. It consists of a DHT lookup service, a trusted identification service, and concentric rings of friend nodes providing trusted data storage. Paths between nodes of adjacent rings are created based on users' trust relationships in real life. Existing paths can therefore be utilized to achieve communication privacy.

Vis-à-Vis [9] is a decentralized OSN framework using virtual individual servers (VISs) [10]. A VIS is a personal virtual server running in a cloud computing infrastructure, e.g., Amazon EC2. All users manage their information in their own VIS and have therefore full control over it. Connections between VISs are realized by DHT lookups.

The concept of personal virtual servers is also used by Diaspora[1]. Diaspora consists of several peer servers (pods) which can securely communicate with each other. A pod can host multiple virtual private servers (seeds), which in turn hosts a user's profile. A hybrid encryption scheme is used for content encryption and key distribution.

Although the former approaches provide some adequate mechanisms for enhancing privacy in OSNs, these mechanisms are only effective in the respective architecture and are hardly applicable to other domains.

C. Global approaches

The following approaches are not restricted to specific services or domains and can be used in a global manner to protect privacy when publishing information online.

[1]https://joindiaspora.com

FireGPG[2] is an Firefox extension providing GnuPG[3] operations on any plaintext in input text-fields of websites. The encrypted ciphertext with additional meta data will be posted to the website. Only a user with the installed plugin and corresponding secret key can decrypt the content. A similar tool is SeGoDoc [11] which provides on-the-fly encryption for Google Docs. The main drawback of such approaches is that a provider can easily detect the ciphertext and delete it or exclude users from its services. Further, the user might not be able to delete the content afterwards.

One approach to achieve transience of published information is Vanish [12]. Vanish allows the creation of encrypted self-destructing data by storing parts of the key in a public DHT. However, Wolchok et al. [13] demonstrated two Sybil attacks against the current Vanish implementation. While Vanish is an interesting approach for giving digital data a lifetime, it still remains hard to control how long the data should persist.

X-pire[4] is another Firefox extension which aims at controlling the lifetime of published pictures. Pictures are encrypted with a symmetric key which is stored with an expiration date on a public server. CAPTCHAs should avoid automatic crawling of keys. However, the effectiveness of X-pire is very limited and its usage can even lead to a decrease of privacy. This was demonstrated by another Firefox extension[5] which automatically collects and stores keys on a separate server.

D. Optimistic Approaches

All of the approaches above assume that receivers of information are not trusted and aim to enforce privacy protection with technical mechanisms. However, enforcing privacy to a full degree is difficult. For instance, even a robust mechanism for controlling access to images will not prevent someone from making a screenshot. Optimistic approaches recognize such limitations and depend on trust in the other party.

The Platform for Privacy Preferences (P3P) [14] provides a standardized mechanism for websites to specify their privacy practices. Those practices can be compared to a user's privacy preferences to indicate compliance or collisions. Approaches that support such privacy indicators in an intuitive way are the Privacy Bird [15] or the concept of privacy nutrition labels [16]. Both approaches are based on simple visual indicators that inform a user about privacy practices of websites.

In addition to indicating websites' privacy practices, it might also be useful to indicate privacy preferences between users. One approach for this are the so-called Privicons[6]. Privicons are a set of icons that communicate a user's privacy expectations in emails and can easily be interpreted by recipients. The adherence cannot be enforced but instead relies on social norms and trust in other persons. The concept has been submitted as an IETF Internet-Draft [17] and was implemented as a Google Chrome extension for Gmail.

[2]http://www.getfiregpg.org
[3]http://www.gnupg.org
[4]http://www.x-pire.de
[5]http://www-sec.uni-regensburg.de/research/streusand
[6]http://privicons.org

III. Requirements

In order to realize a flexible and easy to use online privacy control system, the following requirements must be met:

- **Information Management.** A privacy control system should provide management functionalities for published information. Users should be able to set access control restrictions in terms of privacy policies and be given an overview of information that was already published. Further, users should be able to modify and delete already published information and associated policies. The management of information receivers should support social relations and group associations to enable the formation of flexible and group-based access control policies.
- **Information Security.** Published information must be transmitted and stored securely in order to prevent unintended disclosure or modification. Strong encryption can provide confidentiality of information. Digital signatures by data owners can provide information integrity.
- **Transparency.** The use of a privacy control system should be transparent to service providers and unauthorized persons to prevent penalty and social tensions. The process of access control enforcement should also be transparent for authorized persons, that is no extra interaction should be needed to receive securely published information.
- **Independence.** Users publish their information across different services and platforms online. Therefore, a comprehensive privacy control system should be applicable across platforms and services and should not depend on their cooperation.
- **Usability.** The user's browsing experience should not be influenced significantly by the use of a privacy control system. The support for specifying privacy policies should be as simple and unobtrusive as possible. The reception of accessible information should not need any user interaction at all. However, the user should have the possibility to realize that particular information was shared via the privacy control system.

IV. Privacy Judge

The requirements discussed above led to the realization of our privacy control system *PrivacyJudge*. The basic idea of PrivacyJudge is to decouple information stored on servers of service providers and online platforms from those that should be accessible only to authorized persons. Therefore, PrivacyJudge integrates a privacy control layer between existing content stored on those servers and the information which is actually presented in a user's browser, see Figure 1. Parts of the website's existing content serve as placeholders and implicit or explicit identifiers for sensitive information. Sensitive information is stored encrypted on an external privacy server. A browser-based privacy client supports the user in publishing private information and associates it with real content of a specific website. Further, the privacy client automatically embeds associated information of other users if access has been granted.

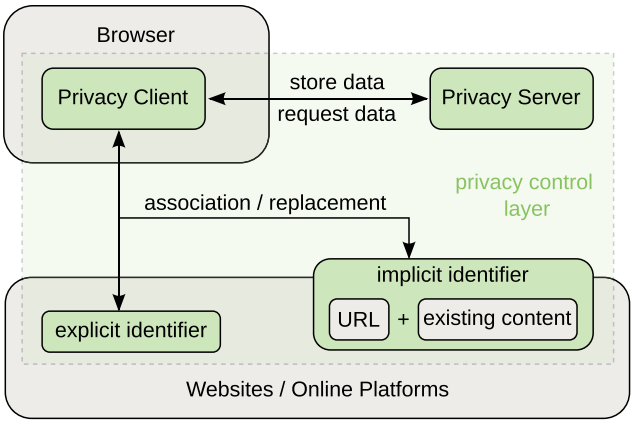


Fig. 1: Conceptual system architecture of PrivacyJudge.

We describe the different architecture components and the main interactions for publishing and receiving information in the following.

A. Setup Procedure

The first step is setting up a PrivacyJudge server or using an existing one provided by a third party. Users need to create an account, which is identified and verified by an unique email address. During the registration process, users choose an account password for authenticating the privacy client to the privacy server. Further, the privacy client creates an asymmetric key pair for information encryption and signing. The account's hashed password and the public key are stored on the privacy server. The account's private key is stored locally on the client.

With the privacy client users can restrict access of published information either to individuals or groups. Therefore, a user can import contacts identified by email addresses and associate them to social groups of similar trust expectations, like *friends*, *family*, or *colleagues*. A contact's public key is fetched from the PrivacyJudge server and stored by the client.

B. Publishing Information

Whenever users want to publish information with PrivacyJudge, they first choose whether the information should be identified explicitly or implicitly, see Figure 2. Explicit identifiers ID_{exp} are automatically generated URIs directly embedded in the website, e.g., by posting into a text field. Explicit identifiers are independent of websites and can be embedded multiple times at different locations. The drawback is that service providers or unauthorized users might recognize these identifiers. Implicit identifiers ID_{imp} depend on a website's specific URL and its DOM structure of some existing content. The user selects the existing content which should be replaced by the new sensitive information. Implicit identifiers cannot be reused at different locations, but they provide complete transparency to service providers and unauthorized users.

After choosing the identifier type, the user inserts the new information and selects contact(s) or group(s) who should have

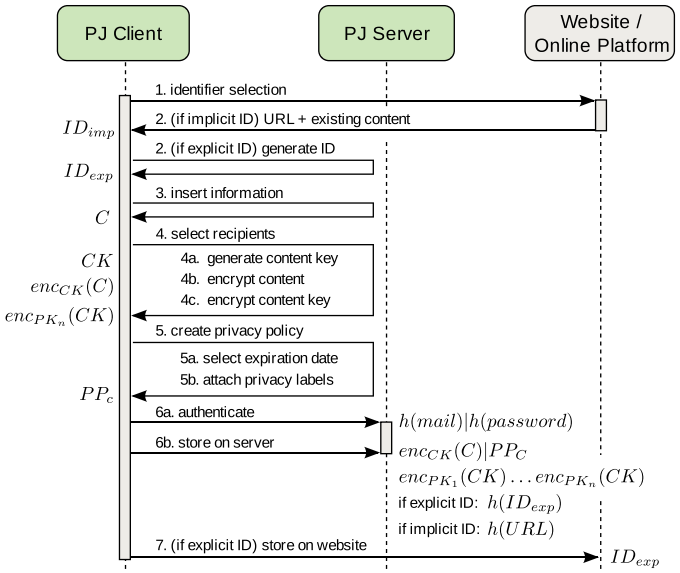


Fig. 2: Basic procedure for publishing information.

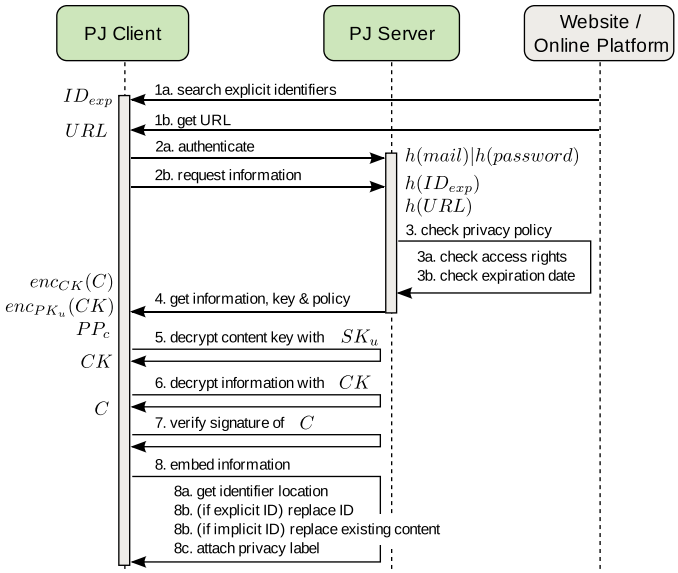


Fig. 4: Basic procedure for receiving information.

Fig. 3: Privicons used for privacy signaling in PrivacyJudge.

access to it. The PrivacyJudge client will generate a symmetric content key CK and encrypt the new information C. If an implicit identifier is used, the identifying existing content and its DOM structure are also encrypted with CK. Optionally, the client can sign the information with the user's secret key SK_u. For each of n recipients the content key CK is encrypted with the recipient's public key PK_n.

The user can further set an expiration date and attach labels, that signal how the information should be treated by recipients. All of these privacy settings form a privacy policy PP_C which is attached to the encrypted content $enc_{CK}(C)$.

Before the client can store information on the PrivacyJudge server, it needs to authenticate itself with the account's hashed email-address and hashed password. After successful authentication the client stores the encrypted content with its policy $enc_{CK}(C)|PP_C$, the encrypted content keys $enc_{PK_1}(CK)\ldots enc_{PK_n}(CK)$, and the hashed identifier on the PrivacyJudge server. For implicit identifiers only a hash of the URL is stored. The server will respect attached expiration dates and automatically purge all expired data.

The concept of attaching privacy labels to signal privacy preferences is an important aspect of privacy protection. Privacy labels rely on social norms and trust in the recipients to properly treat sensitive information. Even if adherence cannot be enforced, it prevents unintentional misuse of information. For privacy labels we use an existing set of privacy icons, called Privicons[7] as depicted in Figure 3. Those icons can signal that a recipient should a) *not attribute* the author, should b) keep information *internal* or c) *private*, can d) *share* information, should e) *not print* or d) *delete it after reading*. Some of the icons can be combined to signal multiple preferences [17], e.g., a) with b), or c) with e).

[7]Designed by A. Braendhaugen for privicons.org, used with permission

C. Receiving Information

On each visited website the PrivacyJudge client checks for explicit identifiers ID_{exp}, see Figure 4. The client authenticates to the PrivacyJudge server and requests associated information for all explicit identifiers and the website's URL by passing $h(ID_{exp})$ or $h(URL)$, respectively. The server checks associated privacy policies and returns the encrypted information $enc_{CK}(C)$ together with the content key $enc_{PK_u}(CK)$ encrypted for the particular user u, if access was granted. The client decrypts CK with its secret key SK_u and decrypts the information C. In case of an explicit identifier the received information will be embedded at the location of the identifier. For implicit identifier, the client searches the identifying existing content attached to C and replaces this content with the new information. The embedded information will be tagged with the associated privacy labels and the author's name if it contains a valid signature, otherwise the author remains anonymous.

D. Content Management

The PrivacyJudge client provides content management capabilities to let users keep track and control all of their published information. Users can easily change or delete information, and also modify associated privacy policies. That is, they can change privacy labels, expiration date, and recipients. Most of these changes require only minor interactions. However, removing recipients requires re-encryption of affected items with a new content key which in turn must be re-encrypted with the public keys of remaining recipients.

V. Implementation

We implemented a prototype of PrivacyJudge client as an extension for the Mozilla Firefox browser. Symmetric content encryption is based on a JavaScript AES implementation[8] with an AES-keylength of 256 bits. For asymmetric encryption[9] of content keys as well as signature[10] creation we utilize a JavaScript RSA library with a RSA-keylength of 2056 bits.

Explicit identifiers, URLs of implicit identifiers, as well as email addresses of user accounts are hashed with a SHA-256 function[11] before transmission to the server.

The current version of the PrivacyJudge client supports publishing of textual information and pictures. In order to publish new information, a user can open the PrivacyJudge dialog via the context menu either from a text input form, from a selected text part, or from an existing picture on the current website. If the dialog (see Figure 5a) is opened from a text input form, the client generates an explicit identifier. Otherwise the website's URL and selected text, or picture URL is used as input for implicit identifiers. Further, the selected HTML element, its parent element, the element's ID and class are used for implicit identifiers and are attached to published information when transmitted to the server.

Whenever the client receives information of other users from the server, it is embedded in the website via JavaScript DOM manipulation. A screenshot of embedded information is depicted in Figure 5b (bottom). A tooltip is showing the author and the attached privacy label.

To simplify the process of privacy policy specifications, a user can define default privacy settings. For example, a default group of recipients, a default expiration date, or default privacy labels. Once a user has published information on a particular website, the information will be listed in the content management dialog of the PrivacyClient, see Figure 5c.

The PrivacyJudge server was realized with PHP and tested with an Apache webserver. User accounts, encrypted information, and privacy policies are stored in a MySQL database. Images are stored encrypted in the file system of the server. Client and server always communicate via HTTPS. Further, the server runs a cronjob to periodically check expiration dates of privacy policies and purge expired information as well as attached encrypted content keys.

VI. Evaluation

A. Security Analysis

We provide a security analysis of PrivacyJudge by discussing several attacks with different intentions against the components of our architecture. We investigate the following adversary profiles and their respective intentions:

- **Service Providers.** Service providers might want to identify users of privacy systems in order to penalize them, e.g., suspend or even delete their accounts.
- **Privacy Server Owners.** In a best case scenario, the privacy server is owned by the user himself or by a trusted third party. However, if this assumption fails, the server owner might try to misuse stored data for own benefits.
- **Users.** Other users of the privacy system might want to know if someone withholds particular information from them or with whom someone shares information.
- **Outsiders.** An external person might want to steal sensitive information from a particular user.

1) Finding published information: PrivacyJudge provides two ways for identifying published personal information: either by an embedded explicit identifier or by a implicit identifier, which includes the URL of the web resource where the private content was published. Therefore, in order find published content an adversary might either try to find embedded identifiers or send requests to the privacy server in order to determine if private content was published for a specific URL.

Finding identifiers requires that the identifier's structure allows an association with PrivacyJudge. For usability reasons, this assumption applies to our prototype. However, our conceptual approach allows the use of arbitrary strings as identifiers. For instance, it is possible to use random URLs of an URL shortener service which would render finding of identifiers much harder. Of course, allowing different structures for identifiers would also increase the complexity for the PrivacyJudge client to find identifiers.

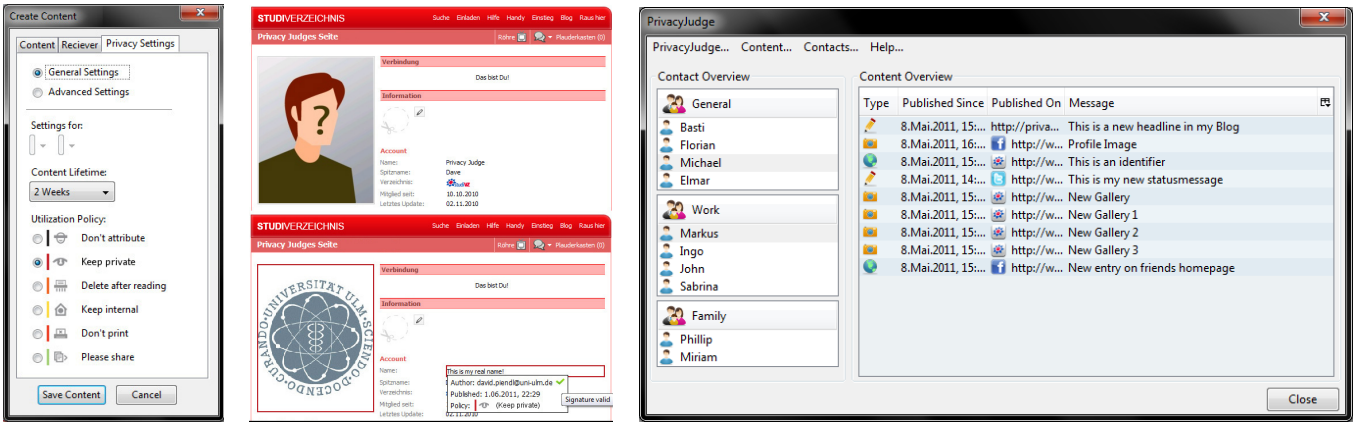
Sending requests to the privacy server in order to find published content for a specific URL is limited to authorized users only. If a request does not provide valid authentication tokens, or an authenticated user does not have access to existing content, the server sends a *404 Not Found* message. This prevents non authenticated as well as non authorized entities from inferring the existence of published content.

The success of the former security mechanisms of course depends on the privacy server's integrity. If an adversary gains access to the server's database, it will be trivial to determine if a particular identifier exists, or if content was published for a particular URL. However, as only the hashes of identifiers or URLs are stored in the database, the reverse direction is not possible. This means it is not possible to infer the publish locations of stored content. Furthermore, it is not possible to infer who published particular content and who has access to it, as account names (i.e., the users' email addresses) are also hashed before they are stored in the database.

2) Accessing published content: In order to gain access to published content, an adversary might attempt different attacks. One attempt is to eavesdrop the connection between privacy client and privacy server. This attack is prevented by using only HTTPS connections. We assume that proper server and client certificates are used.

Another attempt is stealing a user's credentials and private key. The only location to steal those data from is the privacy client. An adversary could use a Trojan to steal this data. To prevent this attack, credentials could be stored in a password safe and the private key in a Trusted Platform Module. However, the current prototype does not provide this functionality.

[8] http://javascript.about.com/library/blencrypt.htm
[9] http://www-cs-students.stanford.edu/ tjw/jsbn/
[10] http://www9.atwiki.jp/kurushima/pub/jsrsa/
[11] http://www.webtoolkit.info/javascript-sha256.html



(a) Policy creation dialog.

(b) OSN site without (top) and with access (bottom) to published information.

(c) Main dialog with contact list and overview of published information.

Fig. 5: Screenshots of the PrivacyJudge Client.

Legitimate receivers of published content also pose a disclosure risk. It is hard to enforce that those users do not further distribute content to unauthorized persons. For instance, one could make a screenshot or print out the contents of a particular website. For this purpose we rely on social norms and privacy signaling through Privicons. The users select who should have access to particular information, therefore already asserting a certain amount of trust to that user. The privacy signaling approach supports the social trust between the entities by reinforcing the intended use of the information.

3) Modifying published content: If an adversary could gain access to the server's database, or the server itself is malicious, one could try to modify published information of other users or try to create new content on behalf of another user. Both scenarios are prevented by our hybrid encryption scheme which provides strong content encryption and the possibility to sign published information with a user's secret key. Recipients can validate information with the user's corresponding public key. Thus, the only way to successfully launch this attack is stealing the user's secret key or replacing that user's public key at all clients of respective recipients. As public keys are already stored in the client during the initial import of a new contact, this is not feasible.

TABLE I: Requirement and feature comparison of existing privacy control approaches

Approaches	*Inf. Management*	*Inf. Security*	*Transparency*	*Independence*	*Usability*	*Privacy Signaling*
NOYB	◐	◐	●	○	◐	○
FaceCloak	◐	◐	●	○	○	○
FlyByNight	◐	◐	◐	○	◐	○
Persona	●	◐	○	○	◐	○
PeerSoN	●	◐	○	○	◐	○
Diaspora	●	●	◐	○	●	○
FireGPG	○	●	○	●	◐	○
SeGoDoc	◐	◐	○	○	◐	○
Vanish	○	◐	○	●	○	○
X-Pire	◐	◐	○	◐	◐	○
Privacy Bird	○	○	●	○	◐	◐
Privicons	○	○	○	●	◐	●
PrivacyJudge	●	●	●	●	◐	●

●: Fulfilled; ◐: Partially fulfilled; ○: Not fulfilled

B. Comparative Analysis

Most of the discussed approaches in Section II provide solutions only for a particular web scenario. For instance, several solutions exist for OSNs, but those are not applicable for other domains. On the other hand, existing global solutions such as the FireGPG or X-Pire plugins are limited in their functionality and only work on websites for unfiltered text or pictures. Furthermore, most solutions are not transparent to service providers or other users, which may have negative effects for a user of a privacy system.

We provide a short comparative analysis of existing approaches and PrivacyJudge with respect to our identified requirements (see Section III) and the privacy signaling feature of PrivacyJudge. The results are summarized in Table I.

- **Information Management.** PrivacyJudge offers an overview of all published sensitive information, allowing to see where it was published, to modify attached policies, or to delete it. Most of the OSN extension solutions depend on the particular OSN management capabilities. Independent OSN solutions like Diaspora or PeerSoN provide their own management functionalities. X-Pire provides the functionality of changing expiration dates of published pictures. However, it is not possible to see where pictures were published.
- **Information Security.** In PrivacyJudge, information confidentiality and integrity are achieved with strong encryption. As far as possible anonymity is provided by the

use of hash functions. Most of the existing approaches do not discuss the use of digital signatures for integrity protection of published information. Only Diaspora and FireGPG do also provide information signing.

- **Transparency**: PrivacyJudge provides complete transparency when using implicit identifiers. FaceCloak and NOYB provide similar transparency by using fake information. Privacy Bird is also transparent to website providers as it only operates on the client side.
- **Independence.** Explicit identifiers of PrivacyJudge can be placed on every website allowing text uploads. Implicit identifiers do also work on any website, but are suited best for websites with low dynamic changes. Only FireGPG and Vanish are independent from any platform specific requirements as well.
- **Usability.** Controlling privacy often introduces additional steps to information publishing. PrivacyJudge provides intuitive dialogs and guides users as much as possible. However, platforms that are designed with privacy in mind from the start, like Diaspora, can provide higher usability for privacy controls.
- **Privacy Signaling.** Signaling privacy preferences to recipients of information is an important privacy mechanism to prevent unintended disclosure which has been neglected by most existing approaches. Even if attached Privicons in PrivacyJudge do not enforce compliance, they rely on social norms to prevent unintentional misuse of information by intended recipients.

VII. Conclusion

PrivacyJudge offers a straightforward approach to effectively controlling privacy online. When posting a new information item, the user selects other users that should have access to the information, how long they should have access, and attaches privacy labels to convey how the data should be treated. Our system combines cryptographic enforcement of access policies with social cues about data usage. By adding privacy labels, the risk of accidental exposure of personal information, by users with access to it, can be reduced.

Our security analysis shows that the PrivacyJudge system remains transparent to the service provider. Only those authenticated users for whom information has been published on the PrivacyJudge server can also retrieve it. Other entities do not gain any information about published content. In comparison with similar privacy control systems PrivacyJudge is more flexible than most approaches, because it is not tailored for a specific service but can be used on arbitrary websites, at least to some extend. From the analyzed systems, only PrivacyJudge actively addresses the issue of how privacy can be protected once an entity is given access to it.

Our prototype underlines the easy setup of PrivacyJudge in form of a browser extension. In addition, a PrivacyJudge server needs to be setup which could be operated by trusted peers. However, the usability of privacy control systems requires further attention. Privacy control systems should naturally integrate into people's online social activities, which is currently not the case. Setup and initialization need to be simplified further to make such services accessible to less tech-savvy users. Also, the overview of information posted online needs to be optimized for accessibility even when dealing with large quantities of information.

We believe that the combination of strong PETs with mechanisms that appeal to human behavior could be a viable path towards improved privacy online and reduce inadvertent disclosure of sensitive information. Interesting work is currently carried out in the areas of social signaling and privacy nudges, especially in the context of online social networks. Exploring further possibilities for integration of such methods with cryptographic PETs will be an interesting challenge for further research.

References

[1] E. Edwards, "Throwing it all away: Community, data privacy and false choices of web 2.0," Santa Clara Univ. Law School, Tech. Rep., 2008.

[2] H. L. Hundley, "US teenagers' perceptions and awareness of digital technology: a focus group approach," *New Media & Society*, vol. 12, no. 3, pp. 417–433, May 2010.

[3] S. Guha, K. Tang, and P. Francis, "NOYB: privacy in online social networks," in *Proc. of the 1st workshop on Online social networks*. ACM, 2008, p. 49–54.

[4] W. Luo, Q. Xie, and U. Hengartner, "Facecloak: An architecture for user privacy on social networking sites," in *Intl. Conference on Computational Science and Engineering*, vol. 3. IEEE, 2009, pp. 26 –33.

[5] M. M. Lucas and N. Borisov, "FlyByNight: mitigating the privacy risks of social networking," in *Proc. of the 7th ACM workshop on Privacy in the electronic society*. ACM, 2008, pp. 1–8.

[6] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin, "Persona: An online social network with user-defined privacy," *ACM SIGCOMM Computer Com. Review*, vol. 39, no. 4, p. 135–146, 2009.

[7] S. Buchegger, D. Schiöberg, L. H. Vu, and A. Datta, "PeerSoN: P2P social networking: early experiences and insights," in *Proc. of the 2nd ACM EuroSys Workshop on Social Network Systems*, 2009, p. 46–52.

[8] L. Cutillo, R. Molva, and T. Strufe, "Safebook: A privacy-preserving online social network leveraging on real-life trust," *Communications Magazine, IEEE*, vol. 47, no. 12, pp. 94–101, 2009.

[9] A. Shakimov, H. Lim, L. P. Cox, R. Cáceres, D. Liu, and A. Varshavsky, "Vis-à-Vis: Privacy-Preserving online social networking via virtual individual servers," in *Proc. of 3rd Intl. Conference on Communication Systems and Networks*. IEEE, 2011.

[10] R. Cáceres, L. Cox, H. Lim, A. Shakimov, and A. Varshavsky, "Virtual individual servers as privacy-preserving proxies for mobile devices," in *Proc. of the 1st ACM Workshop on Networking, systems, and applications for mobile handhelds*, 2009, p. 37–42.

[11] G. D'Angelo, F. Vitali, and S. Zacchiroli, "Content cloaking: preserving privacy with google docs and other web applications," in *Proc. of the 2010 ACM Symposium on Applied Computing*, 2010, p. 826–830.

[12] R. Geambasu, T. Kohno, A. A. Levy, and H. M. Levy, "Vanish: Increasing data privacy with self-destructing data," in *USENIX Security Symposium*. USENIX Association, 2009, pp. 299–316.

[13] S. Wolchok, O. S. Hofmann, N. Heninger, E. W. Felten, J. A. Halderman, C. J. Rossbach, B. Waters, and E. Witchel, "Defeating vanish with low-cost sybil attacks against large DHTs," in *Proc. 17th Network and Distributed System Security Symposium (NDSS)*, 2010, p. 37–51.

[14] L. Cranor, "P3P: making privacy policies more useful," *IEEE Security & Privacy*, vol. 1, no. 6, pp. 50–55, 2003.

[15] L. F. Cranor, M. Arjula, and P. Guduru, "Use of a P3P user agent by early adopters," in *Proc. of the 2002 ACM workshop on Privacy in the Electronic Society*. ACM, 2002, p. 1–10.

[16] P. G. Kelley, J. Bresee, L. F. Cranor, and R. W. Reeder, "A "nutrition label" for privacy," in *Proc. of the 5th Symposium on Usable Privacy and Security*. Mountain View, California: ACM, 2009, pp. 1–12.

[17] U. König and J. Schallaboeck, "Privacy preferences for E-Mail messages," IETF Network Working Group, Internet-Draft draft-koenig-privicons-02, 2011. [Online]. Available: http://tools.ietf.org/html/draft-koenig-privicons-02