A Survey on Privacy and Security in Online Social Networks

Imrul Kayes, University of South Florida
Adriana Iamnitchi, University of South Florida

Online Social Networks (OSN) are a permanent presence in today's personal and professional lives of a huge segment of the population, with direct consequences to offline activities. Built on a foundation of trust–users connect to other users with common interests or overlapping personal trajectories–online social networks and the associated applications extract an unprecedented volume of personal information. Unsurprisingly, serious privacy and security risks emerged, positioning themselves along two main types of attacks: attacks that exploit the implicit trust embedded in declared social relationships; and attacks that harvest user's personal information for ill-intended use. This article provides an overview of the privacy and security issues that emerged so far in OSNs. We introduce a taxonomy of privacy and security attacks in OSNs, we overview existing solutions to mitigate those attacks, and outline challenges still to overcome.

Categories and Subject Descriptors: H.5.2 [**Computer-Communication Networks**] Distributed Systems, Distributed applications

General Terms: Privacy and Security

Additional Key Words and Phrases: Privacy, Security, Online social networks

1. INTRODUCTION

Online Social Networks (OSNs) have become a mainstream cultural phenomenon for millions of Internet users. Combining user-constructed profiles with communication mechanisms that enable users to be pseudo-permanently "in touch", OSNs leverage users' real-world social relationships and blend even more our online and offline lives. As of 2014, Facebook had 1.32 billion monthly active users and it was the second most visited site on the Internet [Alexa, 2014]. Twitter, a social micro-blogging platform, claims over 500 million users [Holt, 2013]. According to Nielsen's 2012 survey, social networking was the fourth most popular online activity [Nielsen, 2012].

Perhaps more than previous types of online applications, OSNs are blending in real life: companies are mining trends on Facebook and Twitter to create viral content for shares and likes; employers are checking Facebook, LinkedIn and Twitter profiles of job candidates[1]; law enforcement organizations are gleaning evidence from OSNs to solve crimes[2]; activities on online social platforms change political regimes [Lotan et al., 2011] and swing election results[3].

Because users in OSNs are typically connected to friends, family, and acquaintances, a common perception is that OSNs provide a more secure, private and trusted internet-mediated environment for online interaction [Cutillo et al., 2009b]. In reality, however, OSNs have raised the stakes for privacy protection because of the availability of an astonishing amount of personal user data which would not have been exposed otherwise. More importantly, OSNs expose now information from multiple social spheres – for example, personal information on Facebook and professional activity on LinkedIn – that, aggregated, leads to uncomfortably detailed profiles [Nissenbaum, 2011].

[1] http://goo.gl/kHJFI5
[2] http://www.cnn.com/2012/08/30/tech/social-media/fighting-crime-social-media/
[3] http://goo.gl/9A6FR

Unwanted disclosure of user information combined with the OSNs-induced blur between the professional and personal aspects of user lives allow for incidents of dire consequences. The news media covered some of these, such as the case of a teacher suspended for posting gun photos [Dam, 2009] or employee fired for commenting on her salary compared with that of her boss [Mail, 2011]), both on Facebook. On top of this, social networks themselves intentionally (e.g., Facebook Beacon controversy [Dwyer, 2011]) or unintentionally (e.g., published anonymized social data used for de-anonymization and inference attacks [Narayanan et al., 2011]) are contributing to breaches in user privacy. Moreover, the high volume of personal data, either disclosed by the technologically-challenged average user or due to OSNs' failure to provide sophisticated privacy tools, have attracted a variety of organizations (e.g., GNIP[4], 80legs[5]) that aggregate and sell user's social network data. In addition, the trusted nature of OSN relationships has become an effective mechanism for spreading spam, malware and phishing attacks. Malicious entities are launching a wide range of attacks by creating fake profiles, using stolen OSN account credentials sold in the underground market [Staff, 2010] or deploying automated social robots [Wagner et al., 2012].

This paper provides a comprehensive review of solutions to privacy and security issues in OSNs. While previous literature reviews on OSN privacy and security are focused on specific topics, such as privacy preserving social data publishing techniques [Zheleva and Getoor, 2011], social graph-based techniques for mitigating Sybil attacks [Yu, 2011], or OSN design issues for security and privacy requirements [Zhang et al., 2010], we address a larger spectrum of security and privacy problems and solutions. First, we introduce a taxonomy of attacks based on OSNs' stakeholders. We broadly categorize attacks as attacks on users and attacks on the OSN and then refine our taxonomy based on entities that perform the attacks. These entities might be human (e.g., other users), computer programs (e.g., social applications) or organizations (e.g., crawling companies). Second, we present how various attacks are performed, what counter-measures are available, and what are the challenges still to overcome.

2. A TAXONOMY OF PRIVACY AND SECURITY ATTACKS IN ONLINE SOCIAL NETWORKS

We propose a taxonomy of privacy and security attacks in online social networks based on the stakeholders of the OSN and the forms of attack targeted at the stakeholders. We identify two stakeholders in online social networks: the OSN users and the OSN itself. On one hand, OSN users share an astonishing amount of information ranging from personal to professional; the misuse of this information can have significant consequences. On the other hand, OSN services handle users' information and manage all users' activities in the network, being responsible for the correct functioning of its services and maintaining a profitable business model. Indirectly, this translates into ensuring that their users continue to happily use their services without becoming victims of malicious actions.

The distinction we make between OSN users and the OSN itself as stakeholders is defined by scale: isolated attacks on users may not affect the wellbeing of the OSN. However, a large attack on user population may translate into reputation damage, service disruption, or other consequences with direct effect on the OSN.

We thus classify online social network privacy and security issues into the following attack categories (summarized in Table I).

(1) Attacks on Users: these attacks are isolated, targeting a small population of random or specific users. We identify various such attacks based on the attacker:

[4]http://gnip.com/
[5]http://80legs.com/

(a) Attacks from other users: Users might put themselves at risk by interacting with other users, specially when some of them are strangers or mere acquaintances. Moreover, some of these users may not even be human (e.g., social robots [Hwang et al., 2012]), or may be crowdsourcing workers strolling and interacting with users for mischievous purposes [Stringhini et al., 2013]. Therefore, the challenge is to protect users and their information from other users.
(b) Attacks from social applications: For enhanced functionality, users may interact with various third-party-provided social applications linked to their profiles. To facilitate the interaction between OSN users and these external applications, the OSN provides application developers an interface through which to access user information. Unfortunately, OSNs put users at risk by disclosing more information than necessary to these applications. Malicious applications can collect and use users' private data for undesirable purposes [Felt and Evans, 2008].
(c) Attacks from the OSN: Users' interactions with other users and social applications are facilitated by the OSN services, in exchange for, typically, full control over user's information published on the OSN. While this exchange is explicitly stated in Terms of Service documents that the user must agree with (and supposedly read first), in reality few users understand the extent of this exchange [Fiesler and Bruckman, 2014] and most users do not have a real choice if they don't agree with the exchange. Consequently, the exploitation by the OSN of user's personal information is seen as a breech of trust, and many solutions have been proposed to hide personal information from the very service that stores it.
(d) De-anonymization and inference attacks: OSN services publish social data for others (e.g., researchers, advertisers) to analyze and use for other purposes. Typically, this data is anonymized to protect user information. However, an attacker can de-anonymize social data and infer attributes that the user did not even mention in the OSN (such as sexual or political orientation inferred from the association with other users).

(2) Attacks on the OSN: these attacks are aimed at the service provider itself, by threatening its core business.
(a) Sybil Attacks: Sybil attacks are characterized by users assuming multiple identities to manipulate the outcome of a service [Douceur, 2002]. Not specific to OSNs, Sybil attacks were used, for example, to determine the outcome of electronic voting [Riley, 2007], to artificially boost the popularity of some media [Ratkiewicz et al., 2011], or to manipulate social search results [Jurek, 2011]. However, OSNs have also become vulnerable to Sybil attacks: by controlling many accounts, Sybil users are illegitimately increasing their influence and power in the OSNs [Yu et al., 2006].
(b) Crawling attacks: Large-scale distributed data crawlers from professional data aggregators exploit the OSN-provided APIs or scrape publicly viewable profile pages to build databases from user profiles and social links. Professional data aggregators sale such databases to insurance companies, background-check agencies, credit-ratings agencies, or others [Bonneau et al., 2009]. Crawling users' data from multiple sites and multiple domains increases profiling accuracy. This profiling might lead to "public surveillance", where an overly curious agency (e.g., government) could monitor individuals in public through a variety of media [Nissenbaum, 2004].
(c) Social Spam: Social spam are contents or profiles that an OSN's "legitimate" users don't wish to receive [Heymann et al., 2007]. Spam undermines resource sharing and hampers interactivity among users by contributing phishing attacks, unwanted commercial messages, and promoting websites. Social spam spreads rapidly via OSNs due to the embedded trust relationships among online friends, which motivates a user to read messages or even click on links shared by her friends.

Attacks on Users	Attacks on the OSN
Attacks from other users	Sybil attacks
Attacks from social applications	Crawling attacks
Attacks from the OSN	Social spam
De-anonymization and inference attacks	Distributed Denial-of-service attacks (DDoS)
	MalwareAttacks

Table I. : Categories of attacks.

(d) Distributed Denial-of-service attacks (DDoS). DDoSes are common forms of attacks, where a service is sent a large amount of seemingly inoffensive service requests that overload the service and deny access to it [Mirkovic et al., 2004]. As many popular services, OSNs are also subjected to such coordinated, distributed attacks.
(e) Malware Attacks: Malware is the collective name for programs that gain access, disrupt computer operation, gather sensitive information, or damage a computer without the knowledge of the owner. ONSs are being exploited for propagating malware [Facebook, 2012]. Like social spam, malware propagation is rapid due to the trust relationships in social networks.

The rest of the paper is organized as follows. Mitigating attacks on users (Sections 3 to 6) include discussions of attacks from other users (Section 3), from social applications (Section 4), from the OSN itself (Section 5), and de-anonymization and inference attacks (Section 6). Mitigating attacks on the OSN (Sections 7 to 11) includes a discussion of Sybil attacks (Section 7), crawling attacks (Section 8), social spam (Section 9), distributed denial-of-service attacks (Section 10) and malware (Section 11). Finally, we conclude the paper in Section 12.

3. MITIGATING ATTACKS FROM OTHER USERS

Users reveal an astonishing amount of personally identifiable information on OSNs, including physical, psychological, cultural and preferential attributes. For example, Gross and Acquisti's study [Gross and Acquisti, 2005a] shows that 90.8% of Facebook profiles have an image, 87.8% of profiles have posted their birth date, 39.9% have revealed phone number, and 50.8% profiles show their current residence. The study also shows that the majority of users reveal their political views, dating preferences, current relationship status, and various interests (including music, books, and movies).

Due to the diversity and specificity of the personal information shared on OSNs, users put themselves at risk for a variety of cyber and physical attacks. Stalking, for example, is a common risk associated with unprotected location information[6]. Demographic re-identification was shown to be doable: 87% of the US population can be uniquely identified by gender, ZIP code and full date of birth [Sweeney, 2000]. Moreover, the birth date, hometown, and current residence posted on a user's profile are enough to estimate the user's social security number and thus expose the user to identity theft [Gross and Acquisti, 2005a]. Unintended revealing of personal information brings other online risks, including scraping and harvesting [Lindamood et al., 2009; Strufe, 2010], social pushing [Jagatic et al., 2007], and automated social engineering [Bilge et al., 2009].

Given the amount of sensitive information users expose on OSNs and the different types of relationships in their online social circles, the challenge OSNs face is to provide the correct tools for users to protect their own information from others while taking full advantage of the benefits of information sharing. This challenge translates into a need for fine-grained settings, that allow flexibility within a type of relationships (as not all friends are equal [Banks and Wu, 2009; Cummings et al., 2002]) and

[6]http://www.theguardian.com/technology/2012/feb/01/social-media-smartphones-stalking

flexibility with the diversity of personal data. However, this fine granularity in classifying bits of personal information and social relationships leads to an overwhelmingly complex cognitive task for the user. Such cognitive challenges worsen an already detrimental user tendency of ignoring settings all together, and blindly trusting the default privacy configurations that serve the OSN's interests rather than the user's.

Solutions to these three challenges are reviewed in the remainder of this section. Section 3.1 surveys solutions that allow fine tunings in setting protection of personal data. The complexity challenge is addressed in the literature on two planes: by providing a visual interface in support of the complex decision that the user has to make (Section 3.2) and by automating the privacy settings (Section 3.3). To address the problem of users not changing the platform's default settings, researchers proposed various solutions presented in Section 3.4.

3.1 Fine-grained Privacy Settings

Fine-grained privacy advocates [Krishnamurthy and Wills, 2008; Simpson, 2008] argue that fine-grained privacy controls are crucial features for privacy management. Krishnamurthy et al. [Krishnamurthy and Wills, 2008] introduce privacy "bits"—pieces of user information grouped together for setting privacy controls in OSNs. In particular, they categorize a user's data into multiple pre-defined bits, namely thumbnail (e.g., user name and photo); greater profile (e.g., interests, relationships and others); list of friends; user-generated content (such as photos, videos, comments and links) and comments (e.g., status updates, comments, testimonials and tags about the user or user content). Users can share these bits with a wide range of pre-defined users, including friends, friends of friends, groups, and all. Current OSN services (e.g., Facebook and Google+) have implemented this idea by allowing users to create their own social circles and to define which pieces of information can be accessed by which circle.

To help users navigate the amount of social information necessary for setting correct fine-grained privacy policies, researchers suggest various ways to model the social graph. One model is based on ontologies that exploits the inherent level of trust associated with relationship definition to specify privacy settings. Kruk [Kruk, 2004] proposes Friend-of-a-Friend (FOAF)-Realm, an ontology-based access control mechanism that uses RDF to describe relations among users. The system uses a generic definition of relationships ("knows") as a trust metric and generate rules that control a friend's access to resources based on the degree of separation in the social network. Choi et al. [Choi et al., 2006] propose a more fine-grained approach, which considers named relationships (e.g., "worksWith", "isFriendOf", "knowsOf") in modeling the social network and the access control. A more nuanced trust-related access control model is proposed by Carminati et al. [Carminati et al., 2006] based on relationship type, degree of separation, and a quantification of trust between users in the network.

For more fine-grained ontology-based privacy settings, semantic rules have been used. Rule-based policies represent the social knowledge base in an ontology and define policies as Semantic Web Rule Language (SWRL) rules. SWRL[7] is a language for the Semantic Web, which can represent rules as well as logic. Researchers used SWRL to express access control rules that are set by the users. Finally, access request related authorization is provided by reasoning on the social knowledge base. Systems that leverage OWL and SWRL to provide rule-based access control framework are [Elahi et al., 2008; Carminati et al., 2009; Masoumzadeh and Joshi, 2011]. Although conceptually similar, [Carminati et al., 2009] provides richer OWL ontology and different types of policies; access control policy, admin policy and filtering policy. A more detailed semantic rule-based model is developed by Masoumzadeh and Joshi [Masoumzadeh and Joshi, 2011]. Rule-based privacy models have two challenges to over-

[7]http://www.w3.org/Submission/SWRL/

come. First, authorization is provided by forward reasoning on the whole knowledge base, challenging scalability with the size of the knowledge base. Second, rule management is complex and requires a team of expert administrators [Engelmore, 1988].

Role and Relationship-Based Access Control (ReBAC) are other types of fine-grained privacy models that employ roles and relationships in modeling the social graph. The working principle of these models is two-fold: 1) track roles or relationships between resource (e.g., photos) owner and the resource accessor; 2) enforce access control policies in terms of the roles or relationships. Fong [Fong, 2011b] proposes a ReBAC model based on the context-dependent nature of relationships in social networks. This model targets social networks that are poly-relational (e.g., teacher-student relationships are distinct from child-parent relationships), directed (e.g., teacher-student relationships are distinct from student-teacher relationships) and tracks multiple access contexts that are organized into a tree-shaped hierarchy. When access is requested in a context, the relationships from all the ancestor contexts are combined with the relationships in the target access context to construct a network on which authorization decisions are made. Giunchiglia et al. [Giunchiglia et al., 2008] propose RelBac, another relation-based access control model to support sharing of data among large groups of users. The model defines permissions as relations between users and data, thus separating them from roles. The entity-relationship model of RelBac enables description logics and as well as the reasoning for access control policies.

In practice, many online social networks (such as Facebook) have already implemented fine-grained controls. A study of Bonneau et al. [Bonneau and Preibusch, 2010] on 29 general purpose online social network sites shows that 13 of them offer a line-item setting where individual data items could be set with different visibility. These line-item settings are granular (one data item is one 'bit') and flexible (users can change social circles).

3.2 View-centric Privacy Settings

Lack of appropriate visual feedback has been identified as one of the reasons for confusing and time consuming privacy settings [Strater and Lipford, 2008]. View-centric privacy solutions are built on the intuition that a better interface for setting privacy controls can impact users' understanding of privacy settings and thus their success in correctly exercising privacy controls. These solutions visually inform the user of the setting choices and consequences of his choices.

In [Lipford et al., 2008], the authors propose an alternative interface for Facebook privacy settings. This interface is a collection of tabbed pages, where each page shows a different view of the profile as seen by a particular audience (e.g., friends, friends of friends, etc), along with controls for restricting the information shared with that group. While this solution provides visual feedback on how other users will see her profile, it's management is tedious for users with many groups.

A simpler interface is proposed by *C4PS* (Colors for Privacy Settings) [Paul et al., 2012], which applies color coding for different privacy visibilities to minimize the cognitive overhead of the authorization task. This approach applies four color schemes for different groups of users; red–visible to nobody; blue–visible to selected friends; yellow–visible to all friends; and green–visible to everyone. A user can change the privacy setting for a specific data item by clicking the buttons on the edge of the attribute. The color of the buttons shows the visibility of the data. If users click "selected friend" (blue) button, a window will open in which friends or groups (a pre-defined set of friends) are granted access to the data item.

A similar approach is implemented in today's most popular OSNs in different ways. For example, Facebook provides a dropdown of viewers (e.g., only me, friends, and public) with icons as visual feedback. In the custom setting, users can set more granular scales, e.g., share the data item with friends of friends, friends of those tagged and restrict sharing with specific people or lists of people.

3.3 Automated Privacy Settings

Automated privacy settings methods employ machine learning to automatically configure a user's privacy setting with minimal user effort.

Fang and Lefevre's *privacy wizard* [Fang and LeFevre, 2010] iteratively asks a user about his privacy preferences (*allow* or *deny*) for specific (*data item, friend*) pairs. The wizard constructs a classifier from these preferences, which automatically assigns privileges to the remaining of the user's friends. The classifier considers two types of features: community structure (e.g., to which community a friend of the user belongs) and profile information (such as age, gender, relationship status, education, political and religion views, work history). The classifiers employed (NaiveBayes, NearestNeighbors and Decision Tree) use uncertainty sampling [Lewis and Gale, 1994], an active learning paradigm, acknowledging the fact that users may quit labeling friends at any time.

Social Circles [Adu-Oppong et al., 2008] is an automated grouping technique that analyzes the users' social graph to identify "social circles", clusters of densely and closely connected friends. The authors posit social circles as uniform groups from the perspective of privacy settings. The assumption is that users will share the same information with all friends in a social circle. Hence, friends are automatically categorized into social circles for different circle-specific privacy policy settings. To find the social circles, they used a (α, β) clustering algorithm proposed in [Mishra et al., 2007]. While convenient, this approach limits users' flexibility in changing the automate settings.

Danezis [Danezis, 2009] aims to infer the *context* within which user interactions happen, and enforces policies to prevent users that are outside that context from seeing the interaction. Conceptually similar to Social Circles, contexts are defined as cohesive groups of users, e.g., groups that have many links within the group and fewer links with non-members of the group. The author used a greedy algorithm to extract the set of groups from a social graph.

An inherent tradeoff for this class of solutions is ease of use vs. flexibility: while the average user might be satisfied with an automatically-generated privacy policy, the more savy user will want more transparency and possibly more control. To this end, the privacy wizard [Fang and LeFevre, 2010] provides for advanced users the visualization of a decision tree model and tools to change it. Another challenge for some of these solutions is bootstrapping: a newcomer in the online social network has no history of interactions to inform such approaches.

3.4 Default Privacy Settings

Studies have shown that users on OSNs often do not take advantage of the privacy controls available. For example, more than 99% Twitter users retained the default privacy setting where their name, list of followers, location, website, and biographical information are visible [Krishnamurthy et al., 2008]. Similarly, the majority of Facebook users has default settings [Gross and Acquisti, 2005b; Acquisti and Gross, 2006; Krishnamurthy and Wills, 2008]. Under-utilization of privacy options are mostly due to poor privacy setting interface [Lipford et al., 2008], intricate privacy settings [Madejski et al., 2011], and inherent trust in OSNs [Acquisti and Gross, 2006; Boyd, 2004]. The problem with not changing the default settings is that they almost always tend to be more open that the users would prefer [Liu et al., 2011]. To overcome this situation, approaches to automatically generate more appropriate default privacy settings have been proposed.

PriMa [Squicciarini et al., 2010] automatically generates privacy policies, acknowledging the fact that the average user will find the task of personalizing his access control policies overwhelming, due to growing complexity of OSNs and the diversity of user content. The policies in PriMa are generated based on the average privacy preference of similar and related users, the accessibility of similar items from similar and related users, closeness of owner and accessor (measured by the number of common

friends), the popularity of the owner (i.e., popular users have sensitive profile items), etc. However, a large number of factors and their parametrized tuning contribute to longer policy generation and enforcement time. A related approach, *PolicyMgr* [Shehab et al., 2010], uses supervised learning of user-provided example policy settings and builds classifiers that are then used for automatically generating access control policies.

Aegis [Kayes and Iamnitchi, 2013a; Kayes and Iamnitchi, 2013b] is a privacy framework and implementation that leverages the *'Privacy as Contextual Integrity'* theory proposed by Nissenbaum [Nissenbaum, 2004] for generating default privacy policies. Unlike the approaches just presented above, this solution does not need user input or access history. Instead, it aggregates social data from different OSNs in an ontology-based data store and then applies the two norms of Nissembaum's theory to regulate the flow of information between social spheres and access to information within a social sphere.

4. MITIGATING ATTACKS FROM SOCIAL APPLICATIONS

Social applications, written by third-party developers and running on OSN platforms, provide enhanced functionality linked to a user profile. For example, Candy Crush Saga[8] (a social game) and Horoscopes[9] (users can check horoscope) are two popular social applications on Facebook.

The social networking platform works as a proxy between users and applications and mediates the communication between them. To better understand this proxy, we show data flow between a third-party social application and the Facebook platform in Figure 1. An application is hosted on a third-party server and runs on user's data that are taken from the Facebook platform. When a user installs the application on Facebook, it takes permission from the user to use some of her profile information. Application developers write the application pages of an application using Facebook mark-up language (FBML)—a subset of HTML and CSS extended with proprietary Facebook tags.

When a user interacts with an application, such as clicks an application icon on Facebook to generate horoscopes (step 1 on Figure 1), Facebook requests the page from the third-party server where the application is actually hosted (step 2). The application requests the user's profile information using secret communication with Facebook (step 3). The application uses the information (e.g., birth date may be used to create horoscopes) and returns a FBML page to Facebook (step 4). Facebook finally transforms the application page from the server by replacing the FBML page with standard HTML, JavaScript (step 5), and transmits the output page to the end user (step 6).

OSN users are facing multiple risks while using social applications. First, an application might be malicious; it could collect a high volume of user data for unwanted usage. For example, to show this vulnerability, BBC News developed a malicious application that could collect large amounts of user data in only three hours [Kelly, 2008].

Second, application developers can violate developer policies to control user data. Application developers are supposed to abide by a set of rules set by the OSNs, called *"developer policies"*. Developer polices are intended to prohibit application developers from misusing personal information or forwarding it to other parties. However, reported incidents [Mills, 2008; Steel and Fowler, 2010] show that applications violate these developer policies. For example, a Facebook application, "Top Friends" enabled everyone to view the birthday, gender and relationship status of all Top Friends users, even though those users kept their privacy for those information to private [Mills, 2008], violating the developer policies that private information of friends are not accessible. The Wall Street Journal finds

[8]https://www.facebook.com/candycrushsaga
[9]https://www.facebook.com/dailyhoroscopes

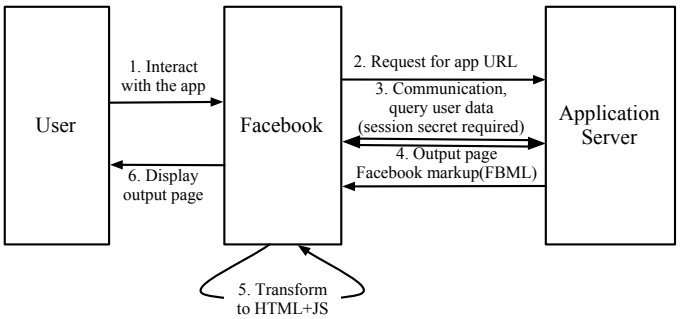

Fig. 1: Data flow in a Facebook application.

evidence that Facebook applications transmit identifying information to advertising and tracking companies [Steel and Fowler, 2010].

Finally, third-party social applications can query more data about a user from an OSN, regardless whether needed or not for proper operation. A study by Felt and Evans [Felt and Evans, 2008] of 150 of the top applications on Facebook shows that most of the applications only needed user name, friends, and their networks. However, 91% of social networking applications have accessed data that they do not need for operation. This violates the principle of least privilege [Saltzer and Schroeder, 1975], which states that every user should only get the minimal set of access rights that enables him to complete his task.

We identified three classes of solutions that attempt to minimize the privacy risks stated above: (i) by anonymizing social data made available to applications (Section 4.1); (ii) by defining and enforcing more granular privacy policies that the third-party applications have to respect (Section 4.2); and (iii) by providing third-party platforms for executing these applications and limiting the transfer of the social data from applications to other parties (Section 4.3).

4.1 Anonymizing Social Data For Third-party Applications

Privacy-by-proxy [Felt and Evans, 2008] uses special markup tags that abstract user data and handle user input. Third-party applications do not have access to users' personal data, rather they use users' IDs and tags to display data to users. For example, to display a user's hometown, an application would use a tag <hometown id="3125"/>. The social network server would then replace the tag with real data value (e.g., "New York") while rendering the corresponding page to the user. However, applications might rely on private data for operations, for example a horoscope application might require users' gender information. A conditional tag handles this dependency (e.g., <if-male> tag can choose the gender of an avatar). Privacy-by-proxy ensures privacy by limiting what applications can access, which might also limit the social value and usability of the applications. Data availability through proxy also means that application developers have to expose the business logic to social network sites (in a form of Javascript to end users). This might discourage third-party developers in the first place. Moreover, applications could still develop learning mechanisms to infer attributes of a user. For example, developers might include scripting code in the personal data dependent conditional execution blocks (if-else) that could send information to an external server when the block executes.

Similar to Privacy-by-proxy, PESAP [Reynaert et al., 2012] provides anonymized social data to applications. However, PESAP secures the information flow inside the browser, so that applications cannot do information leakage though outgoing communications with other third-parties. The anonymization is provided by encrypting the IDs of the entities of the social graph with an application-specific sym-

metric key. Applications use a REST API to get access to the anonymized social graph. PESAP provides a re-identification end-point in order to enable users to see the personal information of their friends in the context of social applications. Secure information flow techniques protect the private information in the browser of a user. This is done by a dynamic, secure multi-execution flow technique [Devriese and Piessens, 2010], which analyzes information flow inside a browser and ensures that the flow complies with certain policies. The multi-execution flow technique labels the inputs and the outputs of the system with security labels and runs a separate sub-execution of the program for each security label. The inputs have designated security labels and can be accessed by a sub-execution having the same or a higher security label. Figure 2 shows the data flow in a PESAP-aware browser.

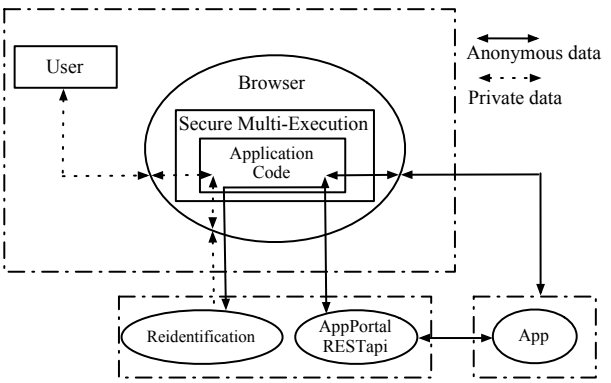

Fig. 2: Data flow in a PESAP aware browser [Reynaert et al., 2012].

4.2 Enforcing Additional Privacy Policies To Social Applications

Besmer et at. [Besmer et al., 2009] propose an access control framework for applications, which adds a new user-application policy layer on the top of the user-user policy to restrict the information applications can access. Upon installing an application, a user can specify which profile information the application could access. However, the framework still uses user-to-user policy to additionally govern an application's access to friends' information on behalf of the user (Alice's installed applications will not get her friend Bob's private data if user-user policy of Bob denies Alice to do so). An additional *friendship-based protection* restricts the information the application can request of a user's friends. For example, Alice installs an application which requests her friend Bob's information and Bob did not install the application. Consider that Bob's default privacy policy is very permissive. But Alice is a privacy conscious and she allows applications to access only the Birth Date attribute. According to friendship-based protection, when the application will request Bob's information via Alice, it will only be able to get Bob's birth date. So, friendship-based protection enables Alice's privacy policies to extend to Bob. The model works well for privacy-savvy concerned users who make informed decisions about an application's data usage while installing an application. An additional functionality could be a set of restrictive default policies for average users.

4.3 Third-party Platforms For Running Social Applications

Egele et al. [Egele et al., 2012] note that, since popular OSN services such as Facebook did not implement user-defined access control mechanisms to date, pragmatic solutions should not rely on the help

of OSNs. They introduce PoX, a browser extension for Facebook applications that runs on a client machine and works as a proxy to provide fine-grained access controls. PoX works as a reference monitor which sits between applications and the Facebook server and controls an application's access to users' data stored on the server. In so doing, an application requests the proxy for users' profile data. Upon receiving the request, the proxy performs access control checks based on user-provided privacy settings. If the request is allowed, the proxy signs the access request with its key, sends the request to the OSN server, and finally replays the result from the server to the application. This application to server data flow is shown in Figure 3. An application developer needs to use the PoX server-side library instead of the Facebook server-side library. One potential challenge is to motivate application developers to write PoX-aware applications when existing mechanisms (e.g., Facebook application environment) are perfectly in place.

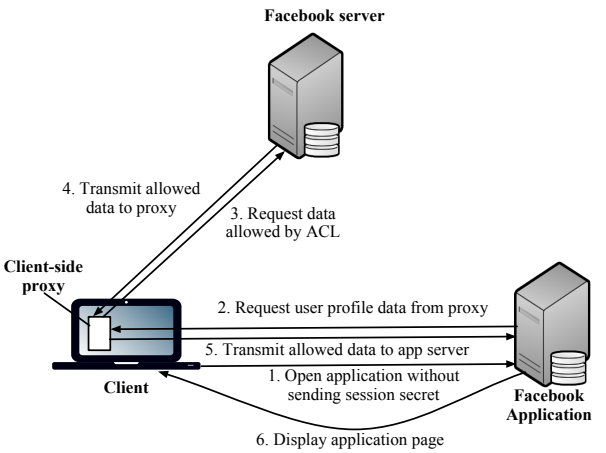

Fig. 3: A data-flow between applications and server with PoX [Egele et al., 2012].

xBook [Singh et al., 2009] is a restricted ADSafe-based JavaScript framework that provides a server-side container in which applications are hosted and a client-side environment to render the applications to users. xBook is different than PoX in that it not only controls third-party applications' access to user data (which PoX also does), but also it limits what applications do with the data. Applications are developed as a set of components; a component is a smallest granular building block of codes monitored by xBook. A component also reveals the information that the component can access and the external entity with which it communicates. During the deployment of an application in xBook, an application developer requires to specify these information. From the specification, xBook generates a manifest for the application. A manifest is a set of statements that specifies what user data the application will use and with which external services it will share the data. At the time of installing the application, the manifest will be presented to the user. In this way, a user will be able to make a more informed decision before installing an application. Although xBook controls third-party applications' access to user data and limits application's data usage, it has to deal with two challenges. First, the platform itself has to be trusted by users and by applications, as it is expected to protect users' personal data and enable third-party applications to execute. Second, hosting and executing applications in xBook requires resources (storage, computation and maintenance) that may be difficult to provide in the absence of a business model.

5. PROTECTING USER DATA FROM THE OSN

The "notice-and-consent" approach to online privacy is the status-quo for practically all online services, OSNs included. This approach informs the user of the privacy practices of the service and provides the user a choice whether to engage in the service or not.

The limitations of this approach have been acknowledged for long. First, the long and abstruse privacy policies offered for reading are virtually impossible to understand, even if the user is willing to invest the time for reading them. For example, on August 2014, we found 4389 words on Facebook's privacy policies and 3473 words on Twitter's privacy policies. Second, such policies always leave room for future modifications; therefore, the user is expected to read them repeatedly in order to practice informed consent. And third, long as they are, these privacy policies tend to be incomplete[10], as they often cannot include all the parties to which user's private information will be allowed to flow (such as advertisers). Consequently, generally people do not read the Terms of Service and when they do, they do not understand them [Fiesler and Bruckman, 2014].

A second serious deterrent for users protecting their online privacy is the "take-it-or-leave-it" "choice" the users are offered. While it may seem as a free choice, in reality the cost of not using the online service (whether email, browsing, shopping, etc) is unacceptably high.

Cornered in this space of falsely informed and lack of choice, users may look for solutions that allow them to use the online service without paying the information cost associated with it. Researchers built on this intuition in two directions. The first direction tends to hide the user information from the very service that stores it (Section 5.1). The second taps into different business models than the ones that make a living from user's private information and replaces the centralized service provider with a fully decentralized solution that is privacy-aware by design (Section 5.2).

5.1 Protection by Information Hiding

This line of work is empirically supported by the Acquisti and Gross's study [Acquisti and Gross, 2006] that shows that while 60% of users trust their friends completely with their private and personal information, only 18% of users trust Facebook to the same degree.

The general approach for hiding information from the OSN is based on the observation that OSNs can run on *fake* data. If the operations that OSNs perform on the fake data are mapped back to original data, users can still use the OSNs without providing them real information. Fake data could be ciphertext (encrypted) or obtained by substituting the original data with pre-mapped data from a dictionary. Encrypted data can be stored on a user's trusted device (including third-party servers or a friend's computer). Access controls are provided by allowing authorized users (e.g., friends) to get the original data from the fake data. Different implementations of this idea are presented next.

flyByNight [Lucas and Borisov, 2008] is a Facebook application that enables users to communicate on Facebook without storing a recorded trace of their communication in Facebook. The flyByNight Facebook application generates a public/private key pair and a password during configuration. The password is used as a key to encrypt the private key and the key is stored on flyByNight server. When a user installs the application, it downloads a client-side JavaScript from the FlyByNight server. This JavaScript does key generation and cryptographic operations. The application knows a user's friends and their public keys who have also installed the flyByNight application. To send messages to friends, a user enters the message into the application and selects the recipient friends. The client-side JavaScript encrypts the content of the message with other users' public keys, tags the encrypted message with the Facebook ID numbers of their recipients, and sends them to a flyByNight message database server. The encrypted messages reside on the flyByNight server. When a user reads a mes-

[10] http://goo.gl/yXqI1s

sage, she provides the password to get the private key (stored in the flyByNight key database). The private key is used to decrypt the message. flyByNight operates under the regulation of Facebook, as it is a Facebook application. It is possible that the computation load on the Facebook servers due to encryption, as well as the suspicious lack of communication among users might attract Facebook's attention and lead to deactivating the application. In the worst case, users lose their ability of hiding their communication, but previous messages remain hidden from the OSN.

Persona [Baden et al., 2009] hides user data from the OSN by combining attribute-based encryption (ABE) and public key cryptography. The core functionalities of current OSNs such as profiles, walls, notes, etc., are implemented in Persona as applications. Persona uses an application "Storage" to enable users to store personal information, and share them with others through an API. Persona application in Facebook is similar to any third-party Facebook application, where users log-in by authenticating to the browser extension. The browser extension translates Persona's special markup language. User information is stored in Persona storage services rather than on Facebook and other Persona users can access the data given that they have the necessary keys and access rights. Similar to the flyByNight, Persona's operation depends on the OSN, as core functionalities are implemented as applications.

NOYB [Guha et al., 2008] distorts user information in an attempt to hide real identities from the OSN, allowing only trusted users (e.g., friends) to get access to the restored, correct information. To implement this idea, NOYB splits a user's information into atoms. For example, Alice's name, gender and age (Alice, F, 26) are split into two atoms: (Alice, F) and (26). Instead of encrypting the information, NOYB replaces a user's atom with pseudorandomly picked another user's atom. So, Alice's first atom is substituted with, for example, the atom (Athena, F) from Athena's profile, and the second atom with Bob's atom from the same class (38). All atoms from the same class for all users are stored in a dictionary. NOYB uses ciphered index of a user's atom to substitute an atom from this dictionary. Only an authorized friend knows the encryption key and can reverse the encryption. A proof-of-concept implementation of NOYB as a Firefox browser plugin adds a button to ego's Facebook profile that encrypts his information and another button on alter's page that decrypts alter's profile. The cleverness of NOYB is that it stores legitimate atoms of information in plain text, thus not raising the suspicions of the OSN. The challenge, however, is the scalability of the dictionaries: the dictionaries are public, contain atoms from both NOYB users and non-users, and are maintained by a third party with unspecified business/incentive model.

FaceCloak [Luo et al., 2009], implemented as a Firefox browser extension, protects user information by storing fake data in the OSN. Unlike NOYB, it does not replace a user's information with another user's information, rather it uses dictionaries and random Wikipedia articles as replacements. A user, say Alice, can protect information from the OSN by using a special marker pre-defined by FaceCloak (@@ in their implementation). When Alice submits the form to the OSN, FaceCloak intercepts the submitted information, replaces the fields that start with the special marker by appropriate fake text and stores the fake data in the OSN. It uses a dictionary (for profile information) and random Wikipedia articles (for walls and notes) to provide fake data. Now, using Alice's master key and personal index key, FaceCloak does the encryption of the real data, computes MAC keys, computes the index, and sends them to a third-party server. Now consider one of Alice's friends Bob, who has installed FaceCloak in his browser, and Bob wants to see Alice's information. After downloading Alice's page (which also includes fake data from the OSN), FaceCloak computes indexes of relevant fields using master and personal index key of Alice. Then it downloads the corresponding values from the third-party server. Upon receiving the value, FaceCloak checks the integrity of the received cipher-text, decrypts it, and substitutes the real data for the fake data. If the value is not found, then the data is left unchanged. See the architecture of FaceCloak in Figure 4. FaceCloak depends on a "parallel" centralized infrastructure to store the encrypted data, which means that a third-party has to maintain all users' data,

probably without getting any benefits from it. And, users have to trust the reliability of the third-party server, which also represents a single point of failure.

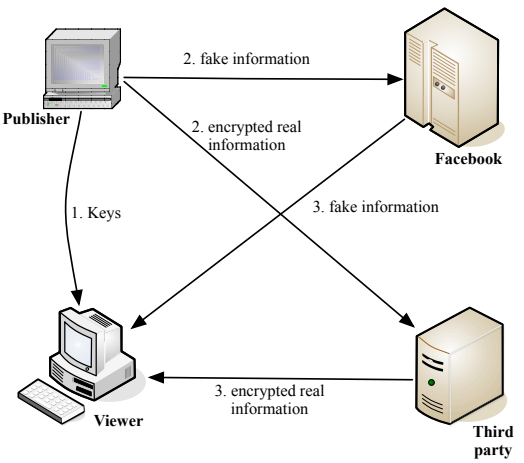

Fig. 4: FaceCloak architecture [Luo et al., 2009].

Virtual Private Social Networks) (VPSN) [Conti et al., 2011], unlike flyByNight, FaceCloak, and NOYB, does not require third-party services to protect users' information from an OSN. Instead, they leverage the computational and storage resources of the OSN users to store real profile data of other users, while storing fake profile data on Facebook. *FaceVPSN* is a Firefox browser extension that implements VPSN for Facebook. In FaceVPSN, user Alice changes her profile information to some fake information and stores the fake information in Facebook and sends by email her correct and fake profiles in a prespecified XML format to her friends. In order to access Alice's real profile, her friends have to have FaceVPSN installed (as a regular Firefox extension) and use its GUI to add Alice's XML file. When Alice's friend Bob requests Alice's Facebook page, Facebook sends an HTML response that has Alice's fake data from Facebook. FaceVPSN's JavaScript code is triggered when "page load" event is fired. The JavaScript code of FaceVPSN searches the profile information of Alice in Bob's stored XML file and replaces the fake information with real information.

Unlike other solutions presented above, FaceVPSN does not risk being suspended by the OSN (since it is not an application running with the OSN's support). Like FaceCloak, however, FaceVPSN requires a user's friends to install the FaceVPSN extension in order to see the user's profile. Moreover, FaceVPSN demands a high degree of user interaction that might affect usability. In particular, upon the addition of a new contact to the friend list, the user has to explicitly exchange profile information with the new friend and upload it into the FaceVPSN application. On top of it, every change of profile information has to be emailed as an XML file to all friends, and the friends are required to go through the XML update process in order to see the changes. This entire process affects usability, given the high number of friends a user might have in OSNs (e.g., half the Facebook users have more than 200 friends, and 15% have more than 500 friends[11])

While the various implementations of the idea of hiding the personal information from the OSN have different tradeoffs, as discussed above, there are also risks associated with the approach itself. First, because the OSN operates on fake data (whether encrypted or randomized), it will not be able to

[11] http://www.pewresearch.org/fact-tank/2014/02/03/6-new-facts-about-facebook/

provide some personalized services such as social search and recommendation. Second, users end up transferring their trust from the OSN to either a third-party server or friends' computers for unclear benefits. The third-party server provides yet another service whose terms of use are probably presented in yet another incomprehensible Terms of Service document, with an opt-out "choice". Friends' computers require extra care for fault tolerance and malicious attacks.

5.2 Protection Via Decentralization

An alternative to obfuscate information from the OSN is to migrate to another service that is especially designed for user privacy protection. Research in this area explored the design space of decentralized (peer-to-peer) architectures for managing user information, thus avoiding the centralized service with a global view of the entire user population. The typical overlay used in most of these solutions is based on distributed hash tables, preferred over unstructured overlays for their performance guarantees. In addition, data is encrypted and only authorized users get access to the plain text. In this section, we discuss decentralized solutions for OSNs. There are three dimensions that differentiate the solutions: (1) how the distributed hash table has been implemented (e.g., OpenDHT, FreePastry, Likir DHT)? (2) where to store users' content (e.g., nodes run by the user, by the friends or cloud infrastructures)? (3) how to manage encryption keys for access controls (e.g., public-key infrastructure, out-of-band)?

PeerSooN's [Buchegger et al., 2009] architecture has two-tiers. One tier, implemented using OpenDHT, serves as a look-up service to find a user. It stores users' meta-data for example, the IP address, information about files, and notifications for users. A peer can connect to another peer asking the look-up service directly to get all required information. The second tier is formed by peers and it contains users' data, such as user profiles. Users can exchange information either through the DHT (e.g., a message is stored within the DHT if receiver of a message is offline) or directly between their devices. The system assumes a public-key infrastructure (PKI) for privacy protection. A user encrypts data with the public keys of the intended audience, i.e., the friends of the user.

Safebook [Cutillo et al., 2009b; Cutillo et al., 2009a] is a decentralized OSN, which uses a peer-to-peer architecture to get rid of a central, omniscient authority. Safebook has three main components: a trusted identification service for certification of public keys and the assignment of pseudonyms; matryoshkas, a set of concentric shells around each user, which serve to replicate the profile data and anonymizes traffic; and a peer to peer substrate (e.g., DHT) for the location of matryoshkas that enables access to profile data and exchange messages.

LifeSocial.KOM [Graffi et al., 2011] is another P2P-based OSN. It implements common functionalities in OSNs using OSGi-based[12] software components called "plugins". As a P2P overlay, it uses FreePastry for interconnecting the participating nodes and PAST for reliable, replicated data storage. The system uses cryptographic public keys as user ID. To protect privacy, a user encrypts a private data object (e.g., profile information) with a symmetric cryptographic key. She then encrypts the symmetric cryptographic key individually with the public keys of authorized users (e.g., her friends) and appends to the data object. The object and the list of encrypted symmetric keys are also signed by the user and they are stored in the P2P overlay. Other users in the system can authenticate the data object by using the public key of the author. But only authorized users (e.g., friends) can decrypt the symmetric key and thus, the content of the object.

LotusNet [Aiello and Ruffo, 2012] is a framework for the implementation of a P2P based OSN on a Likir DHT [Aiello et al., 2008]. It binds a user identity to both overlay nodes and published resources for robustness of the overlay network and secures identity based resource retrieval. Users' information is encrypted and stored in the Likir DHT. Access control responsibility is assigned to overlay index-

[12]http://www.osgi.org/Specifications/HomePage

nodes. Users issue signed grants to other users for accessing their data. DHT returns the stored data to the requestor only if the requestor can provide a proper grant, signed by the data owner.

Vis-a-Vis [Shakimov et al., 2011] targets high content availability. Users store their personal data in Virtual Individual Servers (VISes), which are kept on the user's computer. The server data are also replicated on a cloud infrastructure so that the data is available from the cloud when a user's computer is offline. Users can share information with other users using peer-to-peer overlay networks that connect VISes of the users. The cloud service needs to be rented (considering the high volume of the data users store in OSNs), which makes the scheme monetary dependent.

Prometheus [Kourtellis et al., 2010] is a peer-to-peer social data management system for socially-aware applications. It does not implement traditional OSN functionalities (e.g., profile creation, management, contacts, messaging, etc.), rather it manages users' social information from various sources and exposes APIs for social applications. Users' social data are encrypted and stored in a group of trusted peers selected by users for high service availability. Prometheus architecture is based on Pastry, a DHT-based overlay, and it uses Past to replicate social data. An inference on social data is subject to user defined access control policy enforced by the trusted peers. Prometheus relies on a public-key infrastructure (PKI) for user authentication and message confidentiality.

The toughest challenge for decentralized OSNs is to convince traditional OSN users to migrate to their systems. Centralized social networks have large, established user bases and they are accessible from anywhere. Moreover, they already have a mature infrastructure, making good revenues from users' data and maintaining excellent usability. However, decentralized OSNs are also becoming popular, specially among privacy-aware users. For example, Diaspora[13] is a fully operating open source, stable and decentralized OSN, which relies on user contributed local servers to provide all the major centralized OSN functionalities.

6. MITIGATING DE-ANONYMIZATION AND INFERENCE ATTACKS

Analysis of social data has become immensely popular in a variety of domains, such as biological systems, organizational studies, information science, communication studies, economics, political science, social psychology, development studies, anthropology and sociolinguistics. Researchers and agencies collect or purchase social data to do the analysis. For example, Kwak et al. [Kwak et al., 2010] collected the entire Twitter network as of 2010: 41.7 million Twitter profiles, 1.47 billion follower-following relations, and 106 million tweets. In addition, some organizations and OSN service providers publish social data for others to analyze. For example, the Federal Energy Regulatory Commission published a repository of approximately $500,000$ email messages of Enron Corporation, which has been frequently analyzed for research [Klimt and Yang, 2004; Shetty and Adibi, 2005].

However, publishing and allowing the collection of social network data involves privacy disclosure risks. For example, in 2006 AOL released an anonymized dataset of twenty million search keywords for over 650,000 users [Arrington, 2006]. The dataset was published for research purpose and novel findings emerged (e.g., [Nunes et al., 2008; Adar et al., 2007; Jansen and Booth, 2010]). However, despite the fact that the data released was anonymized, users' privacy was compromised. To make the point, the New York Times identified an individual from this dataset by cross referencing users with phonebook listings.

Privacy attacks in published or collected social network data can be categorized into two categories: de-anonymization attacks and inference attacks. In the following, we discuss the types of de-anonymization attacks, how these attacks take place, and what solutions were proposed to combat such

[13] https://joindiaspora.com/

attacks. More in-depth discussion on de-anonymization can be found in [Zheleva and Getoor, 2011]. In this work, we include the latest work on de-anonymization attacks.

6.1 De-anonymization Attacks

In de-anonymization attacks, an attacker uses external background knowledge and published social data to de-anonymize/identify users in the social graph, and thus learn sensitive user information.

We categorize privacy breaches due to de-anonymization attacks into four classes:

(1) *Identity disclosure* reveals the identity of a user and makes him vulnerable in the real world. For example, although a published dataset on the disease-infection network could advance research on how the disease transmits in communities, an adversary (e.g., an insurance company) that can identify an individual and his disease could exploit this information in unintended ways (for example, for denying insurance).
(2) *Social attributes disclosure* refers to the disclosure of sensitive data associated with a user. For example, disclosure of a user's date of birth, gender and home address could allow the inference of the user's social security number (SSN) and hence could lead to identity theft [Gross and Acquisti, 2005a].
(3) *Relationship disclosure* refers to the situation when the relationships of a user are exposed and this information exploited. For example, two nodes (e.g., companies) in a transaction network are connected by an edge and a weight (e.g., transaction expense) if they are involved in a financial transaction. An adversary, for example a competitor company, can detect whether two target companies have done a financial transaction if it can infer whether an edge exists between the two companies in the network. The adversary can learn the transaction expense from the edge weight and can exploit that information to get advantages.
(4) *Social graph property disclosure* refers to the disclosure of various graph metrics, such as degree, betweenness centrality, closeness centrality, or clustering co-efficient. An attacker can find out the most central users in the network and can make the network structurally vulnerable. For example, an attacker can identify and remove the highest betweenness centrality nodes to disrupt communications between other nodes in the network [Newman, 2010].

6.1.1 *De-anonymization Attack Techniques.* Anonymization is usually done by substituting personally identifying information associated with each user with a random ID [Wu et al., 2010]. However, this substitution is not sufficient to preserve users' privacy. For example, consider a social network in Figure 5(a) that has been anonymized in Figure 5(b) by replacing user names with random IDs. Now, if an attacker knows that Alice, David and Asley are friends of Bob and Alice-David and Asley-David are also friends, a subgraph shown in Figure 5(c), then the attacker can uniquely identify the subgraph in the anonymized network (shown in Figure 5(d)). So, the attacker will be able to re-identify Bob in the anonymized and published social network.

Researchers have shown different techniques to perform de-anonymization attacks. Backstorm et al. [Backstrom et al., 2007] present two types of attacks—*active* and *passive*. In active attacks, the attacker is assumed to be able to modify the network prior to social network data release. The attacker chooses an arbitrary set of target individuals (whose privacy she wants to compromise), creates a small number of new user accounts, makes connections with target individuals (thus forms edges), and establishes a highly distinguishable pattern comprising nodes and edges among the new accounts. The attacker can then efficiently find the subgraph in the released anonymized network, thus can expose the identities of the target individuals.

In passive attacks, an attacker does not have to create new accounts or connections. The intuition is that most nodes in social networks form small uniquely identifiable subgraphs. So, the attacker simply

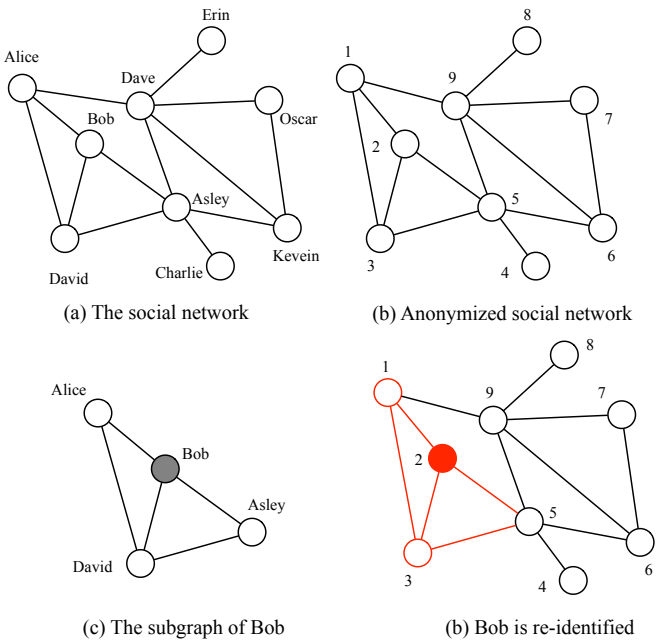

Fig. 5: Anonymization and de-anonmymization attacks.

has to form a coalition with other users. The attacker recruits $k-1$ number of his neighbors and forms a coalition of size k. The users in the coalition know names of their neighbors outside of the coalition. Finally, the attacker tries to identify the subgraph (formed by the coalition) in the published social network, and compromises the privacy of neighboring nodes.

Narayanan and Shmatikov [Narayanan and Shmatikov, 2009] demonstrate the feasibility of a large-scale de-anonymization attack under the assumption that the attacker has background knowledge of a different network whose membership partially overlaps with the target network. Using a de-anonymization algorithm, the authors show that a third of the common users of both Twitter and Flickr can be identified in the anonymous Twitter social graph with a low (12%) error rate.

Both attacks [Backstrom et al., 2007; Narayanan and Shmatikov, 2009] presented above use a subgraph as background knowledge. However, the way an attacker achieves this knowledge is different. In [Backstrom et al., 2007], an attacker creates the background knowledge by adding nodes and edges in the social graphs or forming a coalition among nodes. In [Narayanan and Shmatikov, 2009], the authors propose to collect network data by crawling OSNs, or deploying third-party malicious applications.

6.1.2 *Privacy Preserving Anonymization Methods.* In order to combat de-anonymization attacks, a social network should be anonymized properly before publishing. We categorize privacy preserving social network data anonymization methods into two categories: (1) edge modification-based approaches and (2) clustering-based generalization.

Edge modification-based approaches: Edge modification-based approaches preserve privacy by modifying the social graph structure via addition, deletion or randomization of the edges. Zhour and Pei [Zhou and Pei, 2011] consider a de-anonymization attack, where an attacker, equipped with the

background knowledge about the target's 1-hop neighbors, attempts to re-identify the target in the anonymized dataset using neighborhood matching. Their anonymization method is inspired by k-anonymity model. Although not targeted to social network data publishing, k-anonymity model ensures that each user's information in a released dataset cannot be identified from at least $k-1$ other individuals in the dataset [Sweeney, 2002] .

Zhour and Pei extend k-anonymization to social networks. The goal of the anonymization is to ensure that even knowing the neighborhood of a node, an attacker will not be able to re-identify the node in the anonymized dataset with confidence higher than $\frac{1}{k}$. Let we have a social network $G=(V,E)$ and the anonymized network $\hat{G}=(\hat{V},\hat{E})$, where there exists a bijection function $f:V\rightarrow\hat{V}$ and for each $(u,v)\in E$, $(f(u),f(v))\in\hat{E}$. The authors assume that an attacker has the knowledge of the neighborhood subgraph of a node $u\in V(G)$, denoted by $Neighbor_G(u)$. The goal of the k-anonymization is to ensure that there exist at least $k-1$ other nodes $v_1,v_2,v_3,\ldots,v_{k-1}\in V(G)$ such that $Neighbor_{\hat{G}}(f(v_1)),\ldots,Neighbor_{\hat{G}}(f(v_{k-1}))$ are isomorphic. The anonymization method first extracts the neighborhoods of all nodes in the network. Then it greedily combines nodes into groups and anonymizes the neighborhoods of the nodes in the group, until k-anonymity conditions are met. To anonymize neighborhoods of two nodes such as $Neighbor_G(u)$ and $Neighbor_G(v)$, the method first finds all perfect matches of neighborhood components in $Neighbor_G(u)$ and $Neighbor_G(v)$. For those unmatched components, it tries to pair similar components based on anonymization cost and anonymizes them (this might involve an addition of an edge between two nodes).

Liu and Terzi [Liu and Terzi, 2008] propose k-degree anonymity to combat de-anonymization attacks. They assume that an attacker has the background knowledge of the degree of a target node. The attacker could search the degrees of the nodes in the published network and could re-identify a target node. k-degree anonymity ensures that for every node u in the graph, there exist at least $(k-1)$ other nodes having the same degree as u. Similar to the work of Zhour and Pei, even having the degree background knowledge, an attacker will not be able to re-identify the node in the anonymized dataset with confidence higher than $\frac{1}{k}$. Their anonymization algorithm has two steps. In the first step, it starts from a degree sequence $\mathbf{d}$ of the original network $G(V,E)$ and constructs a new degree sequence $\hat{\mathbf{d}}$ that is k-degree anonymous, so that degree anonymization cost is minimized. In the second step, the algorithm constructs a graph $\hat{G}=(\hat{V},\hat{E})$ such that $\mathbf{d}_{\hat{\mathbf{G}}}=\hat{\mathbf{d}}$, $\hat{V}=V$ and $\hat{E}=E$. To solve the first step, the authors used a dynamic programming method, while the second step is based on a set of graph construction algorithms given a degree sequence with constraints. To construct the new degree sequence the algorithm uses a randomized edge swap transformation strategy.

Clustering-based generalizations: Clustering-based generalizations cluster nodes and edges into groups and anonymize a subgraph into a super-node [Zhou et al., 2008]. So, details about users are hidden.

Hay et al. [Hay et al., 2008] propose a vertex clustering-based generalization approach to combat de-anonymization attacks. They model an attacker's background knowledge as the access to an entity that answers a restricted knowledge query about a target node in the network. They assume three types of queries: *vertex refinement queries* return the local structure of a node in an iterative refined way (e.g., degree of a node, the set of neighbors degrees of a node); *subgraph queries* confirm a subgraph around a target node; *hub fingerprint queries* for a target node returns the vectors of distances between the node and a set of hubs (note that in social networks a hub is defined as a node with high betweenness and degree centrality [Newman, 2010]). The anonymity method is based on structural similarity. The intuition is that structurally similar nodes may be indistinguishable to an attacker. The anonymity method generalizes a social graph by grouping nodes into partitions and publishes the number of nodes in each partition including the densities of edges across and within the partitions. The size of

the partition is at least k (a positive integer), which is similar to k-anonymity in relational data. The method used a simulated annealing algorithm for partitioning [Russell et al., 1995].

Zheleva and Getoor [Zheleva and Getoor, 2008] consider a *link re-identification* attack, where nodes have multiple types of edges and an attacker attempts to re-identify sensitive edges. As a background knowledge, they assume that an attacker can predict a sensitive edge based on other non-sensitive edges. They describe five anonymization techniques: i) remove all sensitive edges; (ii) remove some non-sensitive edges which significantly contribute to the prediction of a sensitive edge; (iii) collapse the anonymized nodes into a single node for each equivalence class (they assume that nodes are clustered into equivalence classes) and publish the count of same types of edges between two equivalence class nodes; (iv) similar technique as (iii), but it needs the equivalence class nodes to have the same constraints as any two nodes in the social network; (v) remove all edges.

Campan and Truta [Campan and Truta, 2009] propose an *edge generalization* technique, leveraging the k-anonymity model. In their model, each node is similar to at least other $(k-1)$ nodes considering attributes and associated structured information (e.g., neighborhood structure of nodes). Nodes are partitioned into clusters and nodes from a cluster are combined into one single node. Edges between two clusters are collapsed into a single edge. An edge between two clusters are labeled with the number of edges between them. While this approach is similar to [Zheleva and Getoor, 2008], two major differences are: (i) [Campan and Truta, 2009] considers all relationships are the same type, but in [Zheleva and Getoor, 2008] there are different types of relations; (ii) [Campan and Truta, 2009] considers both generalization and structural information loss while clustering.

The main challenge for anonymization methods is providing sufficient anonymity while preserving (all) the relevant structural properties of the network. Without preserving enough of the structural properties of the original network, publishing anonymized social network datasets loses its value.

6.2 Mitigating Inference Attacks

The goal of an inference attack is to infer undisclosed private information about a user using other published details of that user. For example, a person might not want to state her political affiliation in Facebook because of privacy concerns. But if he is a member of "ban the same sex marriage" group, then from this group membership an inference may be possible regarding his political affiliation.

Zheleva and Getoor [Zheleva and Getoor, 2009] study four social networks (Facebook, Flickr, Dogster and BibSonomy) and show how an attacker can exploit public and private user profiles to learn private attributes such as user location and gender. They show that declared social relationships and inferred group memberships are enough to predict undisclosed private information. Using the classification model *LINK-GROUP*, a combination of link and group-based classification models, they were able to accurately discover the information of private-profile users.

Heatherly et al. [Heatherly et al., 2013] describe three sanitization techniques to prevent undisclosed private information inference from a released social network dataset. First, they build classification models to accurately predict private data from the available details (attributes) of a user. Then they apply the sanitization techniques to reduce the accuracy of the models. In brief, the techniques are as follows: (i) remove some details (e.g., attributes) to decrease the classification accuracy of sensitive attributes; (ii) alter the link structure of the social graph by adding and removing links and (iii) provide a generalization of details. For example, if a user inputs a favorite activity as "Boston Celtics", the name will be replaced by a more generalized term "Basketball". Experimenting on a Facebook dataset, the authors conclude that removal of attributes and friendship links together in the published data is the best way to reduce classifier accuracy.

Dey et al. [Dey et al., 2012] attempt to infer the age of over one million Facebook users in New York city. Exploiting the Facebook social graph, they design an iterative algorithm which estimates a

user's age based on her friends' ages (e.g., from inferred high school graduation year), friends of friends' edges and so on. They find that for most users, including users who take maximal measures to prevent privacy leakage by hiding their friend lists, it is possible to estimate ages with an error of only a few years. The authors recommend to hide high school graduation year and friend lists to other users who are not friends from users' profiles as a solution.

7. MITIGATING SYBIL ATTACKS

The Sybil attack is a fundamental problem in distributed systems. The term *Sybil* was first introduced by Douceur [Douceur, 2002], inspired from a 1973 book after the same name about the treatment of a person Sybil Dorsett, who manifests sixteen personalities. In Sybil attacks, an attacker creates multiple identities and influence the working of the system.

OSNs including Digg, YouTube, Facebook and BitTorrent have become vulnerable to Sybil attacks. For example, Facebook anticipates that up to 83 million of its users may be illegitimate [BBC, 2012], which is far more than what it anticipates (54 million) earlier [Cellan-Jones, 2012]. The high number of Sybils in the OSN is due to the fact that users can create accounts on OSNs quickly and freely; typically only an email address is enough for an account opening.

Sybil users affect the correct functioning of the system by contributing malicious contents. By controlling a lot of identities, Sybil users increase their influence and power in the OSNs [Nazir et al., 2010; Ratkiewicz et al., 2011]. For example, Sybil users can outvote real users on YouTube and can promote clients' content to the top position by giving more up votes [Riley, 2007]. In Facebook, users control Sybil identities to gain higher status in social games [Nazir et al., 2010]. In Twitter, paid organizations conduct political campaigns disguising themselves as mass population [Ratkiewicz et al., 2011]. Researchers find that Sybils forward malware and spam on social media [Gao et al., 2010; Grier et al., 2010; Irani et al., 2010; Stringhini et al., 2010; Thomas et al., 2011]. Sybil identities are used to acquire users' private contact lists [Bilge et al., 2009; Fong, 2011a]. Moreover, Sybils manipulate Google social search results [Jurek, 2011] and location crowdsourcing results [Marinando, 2010].

Malicious activities from Sybil users are posing serious threats to OSN users, who trust the service and depend on it for online interactions. In future the threat will be aggravated as nowadays more people are relying on OSNs for primary online communications [Lenhart et al., 2010; Murphy, 2010] and ready-to-get news [Kwak et al., 2010]. Sybils cost OSN providers, too, in terms of monetary losses and time. OSN providers spend significant resources and times to detect, verify, and shut down Sybil identities. For example, Tuenti, the largest OSN in Spain, dedicates 14 full-time employees to manually verify user reported Sybil identities [Cao et al., 2012].

However, mitigation of the Sybil attack is not easy. Some open systems employ CAPTCHA [von Ahn et al., 2004], IP address filtering, and computational puzzles to mitigate Sybil attacks [Walsh and Sirer, 2006; Peterson and Sirer, 2009; Piatek et al., 2008]. Unfortunately, none of the solutions worked well in a real-world system. Crowdsource CAPTHA cracking businesses [Acohido, 2009] employ cheap laborers from the undeveloped countries to break codes. The IP address filtering allows only one account/identity per IP address, which causes serious problems for users behind NATs, and yet fails against the attackers controlling a subnet [Yu et al., 2006]. And finally, computational puzzles require a potential OSN registration to solve a computational problem with some controlled difficulty. Although it requires human efforts, an attacker might be equipped with more resources and could solve the problem. Moreover, all of these solutions cannot reduce the number of Sybil identities in the system, at best they can limit the rate of Sybils' intrusion in the system.

Two categories of solutions are available to defend Sybils: Sybil detection and Sybil resistance. Sybil detection schemes [Yu et al., 2006; Cao et al., 2012; Wang et al., 2013; Yang et al., 2011] leverage the social graph structure to identify whether a given user is Sybil or non-Sybil (Section 7.1). On the other

hand, Sybil resistance schemes do not explicitly label users' as Sybils or non-Sybils, rather they use application-specific knowledge to mitigate the influence of the Sybils in the network [Post et al., 2011; Post et al., 2011; Viswanath et al., 2012b] (Section 7.2). In a tutorial and survey Haifeng Yu [Yu, 2011] compiles social graph-based Sybil detection techniques. In this paper, we report latest works on that category, as well as Sybil resistance schemes.

7.1 Sybil detection

Sybil detection techniques model an online social network (OSN) as an undirected graph $G = (V, E)$, where a node $v \in V$ is a user in the network and an edge $e \in E$ between two nodes corresponds to a social connection between the users. This connection could be a friendship relationship on Facebook or a colleague relationship on LinkedIn, and is assumed to be trusted.

The social graph has $n = |V|$ nodes and $m = |E|$ edges. By definition, if all nodes correspond to different persons, then the system should have n users. But, some persons have multiple identities. These users are Sybil users and all the identities created by a Sybil user are called *Sybil identities*. An edge between a Sybil user and a non-Sybil user may exist if a Sybil user is able to create a relationship (e.g., friend, colleague) with a non-Sybil user. These types of edges are called *attack edges* (see Figure 6).

Attackers can launch Sybil attacks by creating many Sybil identities and creating attack edges with non-Sybil users. Detection systems against Sybil attacks provide mechanisms to detect whether a user (node) $v \in V$ is Sybil or non-Sybil. Those mechanisms are based on the authority (e.g., the OSN provider) knows the topology of the network (a centralized solution), or a node only knows its social connections (a decentralized solution). Some common assumptions of Sybil detection schemes are below.

Assumption 1: Attackers can create a large number of Sybil identities in OSNs and can create connections among those Sybil identities, but they lack trust relationships because of their inability to create an arbitrary number of social relationships to non-Sybil users. Intuitively, a social relationship reflects trust and an out-of-band social interaction. So, it requires significant human efforts to establish such a relationship. The limited number of attack edges differentiates Sybil and non-Sybil regions in a social graph as shown in Figure 6.

Assumption 2: The non-Sybil region of a social graph is fast-mixing. Mixing time determines how fast a random walk's probability of landing at each node reaches the stationary distribution [Boyd et al., 2005; Flaxman, 2008]. A limited number of the attack edges causes sparse cut between Sybil and non-Sybil regions. Non-Sybil regions do not show sparse cut as non-Sybils are well connected. As such, there should be a difference in terms of mixing time of the non-Sybil regions compare to the entire social graph.

Assumption 3: The defense mechanism knows at least one non-Sybil. This assumption is essential in a sense that without this knowledge the Sybil and non-Sybil regions become identical to the system.

Most of the Sybil detection techniques are based on social graphs. Social graph-based approaches leverage random walks [Yu et al., 2006; Danezis and Mittal., 2009; Cao et al., 2012; Wei et al., arch], social community [Viswanath et al., 2010], and network centrality [Xu et al., 2010a] to detect Sybils in the network. SybilGuard [Yu et al., 2006] is a decentralized Sybil detection scheme, which uses Assumption 1, Assumption 2 and Assumption 3. A social graph with a small quotient cut has a large mixing time, which implies that a random walk should be long in order to converge to the stationary distribution. So, the presence of too many Sybil nodes in the network disrupts the fast mixing property, in a sense that they increase social network mixing time by contributing small quotient cuts. Thus, a verifier, which is itself a non-Sybil node, can break this symmetry by examining the anomaly of the mixing time in the network. In order to detect Sybils, a non-Sybil node (say a verifier) can perform a random route starting from itself and of a certain length w (a theoretically identifiable quantity, but

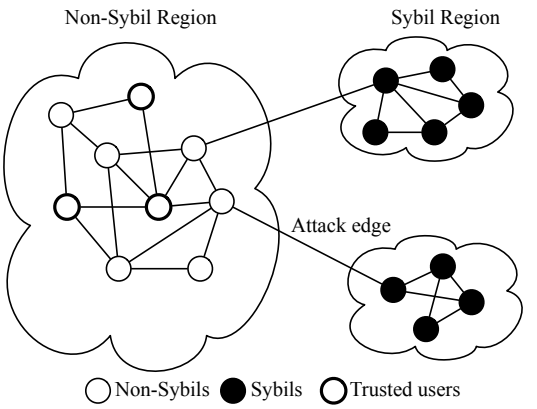

Fig. 6: The system model for Sybil detection.

the paper experimentally shows that this is 2000 for a topology of one-million nodes). A suspect (a node that is in question) is identified as non-Sybil if it's random route intersects with the verifier's random route. As the underlying assumption is that the number of attack edges should be limited, the verifier's route should remain within the non-Sybil region with high probability, given the appropriate choice of w.

SybilInfer's [Danezis and Mittal., 2009] assumptions are also Assumption1, Assumption 2 and Assumption 3. Moreover, it assumes that a modified random walk over a social network, that yields a uniform distribution over all nodes, is also fast mixing. The core of SybilInfer is a Bayesian inference that detects approximate cuts between non-Sybil and Sybil regions in social networks. These identified cuts are used to infer the labels (Sybil or non-Sybil) of the nodes, with an associated probability.

SybilRank [Cao et al., 2012] is also a random walk-based Sybil detection scheme, which uses all three assumptions and ranks user according to their perceived likelihood of being Sybils. Using early terminated power iteration, SybilRank computes landing probability of random short walks and from that it ranks users, so that substantial portion of the Sybil users have low rank. The design of SybilRank is influenced by an observation on early terminated random walks in social graphs—if a walk of this kind starts from a non-Sybil node, then it has a high degree-normalized landing probability to land at non-Sybil node than a Sybil node. SybilRank terms the probability of a random walk to land on a node as the node's *trust*, ranks nodes based on that and filters lower ranked nodes as potential Sybil users. Rather than keeping computationally intensive a large number of random walk traces used in other graph-based Sybil defense schemes [Yu et al., 2006; Yu et al., 2010], it uses power iteration [Langville and Meyer, 2004] in calculating the landing probability of random walks.

Operations performed by SybilRank are shown in Figure 7.

Viswanath et al. [Viswanath et al., 2010] suggest to use community detection algorithms for Sybils' detection. They show that although other graph property based Sybil defense schemes have different working principles, the core of those works revolves around detecting local communities around a trusted node. So, existing community detection algorithms could be used to defend the Sybils also. Although, not explicitly mentioned, their approach is centralized, because community detection requires a central authority to have the knowledge of the entire topology.

Xu et al. [Xu et al., 2010a] propose Sybil detection based on the betweenness rank of the edges. The betweenness of an edge is defined as the number of shortest paths in the social graph passing the edge [Brandes, 2001]. The scheme assumes that the number of attack edges is limited and Sybil and

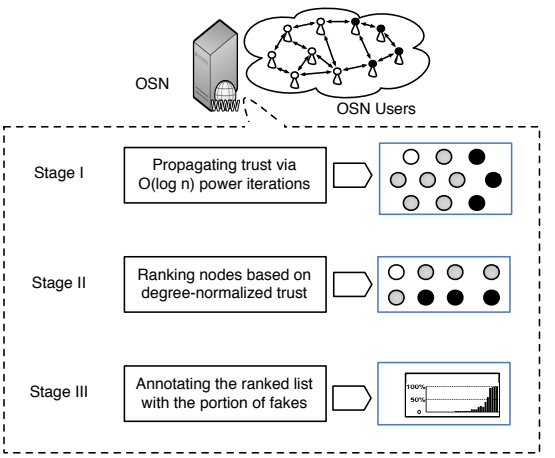

Fig. 7: Three steps performed by SybilRank in detecting Sybils. Black users are Sybils ([Cao et al., 2012]).

non-Sybil regions are separate clusters of nodes. So, intuitively betweenness scores of the attack edges should be high as they connect the clusters. Their scheme exploits this social network property and uses a Sybil Resisting Network Clustering (SRNC) algorithm to detect Sybils. The algorithm computes the betweenness of each edge and identifies the edges with high betweenness as attack edges.

Social graph based approaches still have some challenges to overcome. First, as graph-based Sybil detection schemes exploit trust relations, the success of the identification highly depends on the trust related assumptions. If an assumption is not right in a network, social graph-based Sybil detection techniques might work poorly in that network. For example, the assumption that Sybils' have problems in creating social connections with legitimate users (non-Sybils) is not well established. Although study [Motoyama et al., 2011] shows that most of a Sybil identity's connections are also Sybil identities and Sybils' have less relationships with non-Sybil users, several other studies [Boshmaf et al., 2011; Irani et al., 2011; Bilge et al., 2009] show that users are not careful while accepting friendship requests and Sybil identities can easily befriend with them. Moreover, Sybil users are using advanced techniques to create more realistic Sybil identities, either by copying profile data from existing accounts, or by assigning real users to customize them. Also, another assumption that a social network is fast-mixing may not be right for all social networks. Study [Mohaisen et al., 2010] shows that many of the social networks are not fast-mixing, especially where edges represent strong real-world trust (e.g., DBLP, Epinions, etc.).

Second, the performance of random walk-based Sybil detection techniques depends on the various relevant parameters of the random walks (e.g., the length of a random walk). These factors will work for a fixed network size (as all the schemes have shown), but they have to be updated with the evolution of the social networks.

7.2 Sybil Resistance

Sybil resistance schemes do not explicitly label users' as Sybils and non-Sybils, rather they attempt to mitigate the impact that a Sybil user can have on others. Sybil resistance schemes have been effectively used in applications from diverse domains including content rating systems [Tran et al., 2009; Chiluka et al., 2012], spam protection [Mislove et al., 2008], online auctions [Post et al., 2011], reputation systems [DeFigueiredo and Barr, July], and collaborative mobile applications [Quercia and Hailes, 2010].

Note two assumptions of Sybil detection schemes: 1) non-Sybil region is fast mixing, 2) Sybils can not create an arbitrary number of social relationships with non-Sybils. Sybil resistance schemes also assume that non-Sybils' have a limited number of social connections, but they do not rely on the fast mixing nature of the non-Sybil regions. However, Sybil resistance schemes take an additional application related information such as users' interactions/transactions/votes etc. Using the underlying social network of the users and system information, Sybil resistance schemes determine whether an action performed by a user should be allowed or denied.

Most of the Sybil resistance schemes [Mislove et al., 2008; Post et al., 2011; Viswanath et al., 2012b] share a common approach in resisting Sybils—they use a *credit network* built on the top of the social network of users [Viswanath et al., 2012a]. Originally proposed in the electronic commerce community, *Credit Networks* [Dandekar et al., 2012; Ghosh et al., 2007] create mutual trust protocols in a situation where there is pairwise trust between two users, and a centralized trusted party is unavailable. Nodes in a credit network trust each other by providing credits up to a certain limit. Nodes use these credits to pay for services (e.g., sending a message, purchase items, vote casting) that they receive from one another. These schemes assign credits to the network links, and allow an action between two nodes if there is a path between them that has enough credit to satisfy the operation. As such, these schemes find a credit assignment strategy in the graph and apply the credit payment scheme to allow a limited number of illegitimate operations in the system. A Sybil user has limited number of edges with non-Sybils (hence, limited credits available), which restricts her to gain additional advantages by creating multiple Sybil identities. This scenario is shown in figure 8, which is a core defense philosophy of some resistance schemes. In the following, we provide a brief overview of the Sybil resistance schemes.

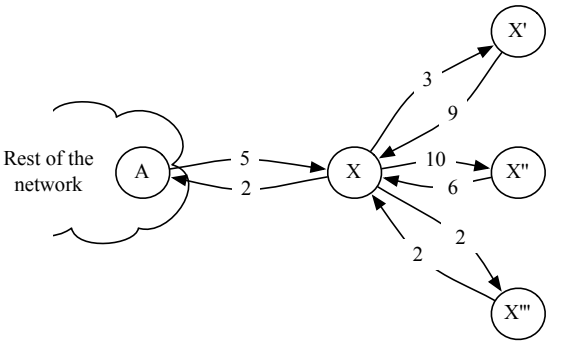

Fig. 8: Credit network based Sybil resistance [Viswanath et al., 2012a]. The network contains four Sybil identities as nodes X, X', X'', X''' of a Sybil user. A directed edge (X,Y) represents how much credit is available to X from Y. If X wants to pay credits from other three nodes, the credits must be deducted from X's single legitimate link to A. So, a Sybil's other identities do not provide any additional credits in the rest of the network.

Ostra [Mislove et al., 2008] leverages existing trust relationships among users to thwart unwanted communication (e.g., spam). It bounds the total number of unwanted communications a Sybil user can produce by assigning credit values to the trust links. If a user sends a message to another user, Ostra finds a path with enough credit from the sender to the receiver. If a path is available, credit is assigned along all the links in the path, which is refunded if the receiver considers the messages as not unwanted. However, if no such path exists, Ostra blocks the communication, but the credit is paid. In this way, Ostra ensures that a user with multiple identities cannot send a large number of unwanted communications, unless she also has additional trust relationships.

Bazaar [Post et al., 2011] is targeted to strengthen the users' reputation in online marketplaces like eBay. The opportunity to create accounts freely leads Sybil users to create multiple accounts and causes the waste of time and significant monetary losses for defrauded users. To mitigate Sybil users, Bazaar creates transaction network by linking users who have made a successful transaction. The weight of a link is the amount that has been successfully transferred due to the transaction. Prior to a transaction, using a max flow based technique, Bazaar computes the reputation of the users doing the transaction and compares with the new transaction value. If it finds available flow, it removes the value of the transaction between the users as credits, and eventually adds back if the transaction is a fraud. However, a new transaction is denied if essential flow is not found.

Canal [Viswanath et al., 2012b] complements Ostra and Bazaar credit networks-based Sybil resistance schemes by applying landmark routing-based techniques in calculating credit payments over a large network. One of the major problems of Ostra and Bazaar is that they require computing max-flow over a graph. However, the huge size of present day network (Facebook has over billion of nodes in social graph) leads to significant computation complexity to compute the max-flow between two nodes in the network. As such, this poses a bottleneck to those techniques to practically deploy in a real-world social network. Canal efficiently computes an approximate max-flow (compromising accuracy with speed-up) path using existing landmark routing-based algorithm [Tsuchiya, 1988; Gubichev et al., 2010]. The main components of Canal are *universe creator processes* and *path stitcher processes*. Universe creator processes continuously select new landmarks and path stitcher processes continuously process incoming credit payment requests. Using real-world network datasets the authors show that Canal can perform payment calculations efficiently (within a few milliseconds), even if the network contains hundreds of millions of links.

MobID [Quercia and Hailes, 2010] makes co-located mobile devices resilient to Sybil attackers. Portable devices in close proximity of each other could collaborate various services (e.g., run localization algorithms to get a precise street map), which is severely disrupted by Sybil users (Sybils could inject false information). MobID uses mobile networks to Sybil resilience. More specifically, a device manages two small networks as it meets with other devices. A network of friends contains non-Sybil devices and a network of foes contains suspicious devices. Using two networks, MobID determines whether an unknown device is attempting a Sybil attack. MobID ensures that a non-Sybil device accepts, and accepted by most other non-Sybil devices with high probability. So, a non-Sybil device could successfully trade services with other non-Sybil devices.

8. MITIGATING ATTACKS FROM LARGE-SCALE CRAWLERS

OSNs enhance social browsing experience by allowing users to view public profiles of others. This way a user meets others, gets a chance to know strangers and eventually befriends some of them. Unfortunately, attackers are there in the vast landscape of OSNs, who exploit this functionality. Users' social data are always invaluable to marketers. Professional data aggregators build databases using public views of profiles and social links and sale the databases to insurance companies, background-check agencies and credit-ratings agencies [Bonneau et al., 2009]. For example, crawling 100 million public profiles from Facebook created news recently [Bradley, 2012]. Sometimes crawling is a violation of terms of service. Facebook states that someone should not collect "...users' content or information, or otherwise access Facebook, using automated means (such as harvesting bots, robots, spiders, or scrapers) without our prior permission" [Facebook, 2015].

One solution of the problem could be the removal of the public profile view functionality. But removal of the public profile view functionality is against the business model of OSNs. Services like search and targeted advertisements bring new users and ultimately revenues to OSNs, but openly accessible contents are necessary for their operation [Wilson et al., 2010]. Moreover, removal of the public view

functionality will undermine user experience, as it makes a connection, communication and sharing easy with unknown people in the network.

OSN operators such as Facebook and Twitter attempt to defend large-scale crawling by limiting the number of user profiles a user can see from an IP address in a time window [Stein et al., 2011]. However, tracking users with low level network identifiers (e.g., IP address, TCP port numbers or SSL session IDs) is fundamentally flawed as a solution of this problem [Wilson et al., 2010]. Aggressive attackers may gather a large vector of those identifiers by creating a large number of fake user accounts, gaining access to compromised accounts, virtualizing in a cloud, employing botnets, and forwarding requests to proxies. Until now, researchers have leveraged encryption based technique [Wilson et al., 2010] and crawler's observational behavior [Mondal et al., 2012] to combat the problem.

ONS's anti-crawling techniques suffer from the fact that web clients can access a particular page using a common URL accessible to all clients [Wilson et al., 2010]. This can be exploited by a distributed crawler e.g., a crawling thread can download and parse a page for links using a session key and can deliver those links to another crawling thread to download and parse using different session keys. So, if some crawlers get banned from the OSNs for malicious activities, the links they have parsed are still valid and a fresh start is possible from those links. SpikeStrip [Wilson et al., 2010] overcomes the problem by creating unique, per-session "views" of the protected website that forcibly tie each client to their session keys. SpikeStrip is a web server add-on that leverages link encryption technique. It allows OSN administrators to moderate data access, and it defends against large-scale crawling by securely identifying and rate limiting individual sessions.

When a crawler visits a page, it receives a new session key and a copy of the page whose links are all encrypted. SpikeStrip appends each user's session key to those links and then encrypts the result using a server-side, secret symmetric key. It also appends a *salt* to the link after encryption to make each link unique. As time passes, the crawler progressively covers more pages and collects links. However, at a point, the crawler requires to change the session key due to the expiration of the session or due to a ban from the OSN. As SpikeStrip couples all URLs to the browser's session key, this switching of sessions invalidates all the links collected for future traversals. Thus, a fresh start to reconstruct the collection should be started from the beginning. The authors implemented mod_spikestrip, a SpikeStrip implementation for Apache 2.x and showed that it imposes only 7% performance penalty on Apache.

Genie [Mondal et al., 2012] exploits browsing patterns of honest/real users and crawlers and thwarts large-scale crawls in OSNs using Credit Networks [Dandekar et al., 2012; Ghosh et al., 2007]. The system design is based on three observations from real-world datasets: (i) there is a balance between the number of profiles a honest user views and views requested by other users to her profile, but crawlers view many more profiles than the number of times their profiles are viewed; (ii) a honest user views profiles of socially close users.(iii) a honest user repeatedly views a small set of profiles in the network, but unless re-crawling, the crawlers avoid repeating viewing of other users' profiles. Genie leverages these observations and enforces a viewer to make a "credit payment" in the credit network if a user wants to view a profile. It allows a user (also might be a crawler) to view a profile if a max-flow between them has at least a threshold value. The required credit payment to view a profile depends on the shortest path length from viewer to viewee; a user has to pay more to view the profile of a distant user in the social graph. As a legitimate user usually views one or two hop distant profiles, and also other users also view her profile, her liquidity of credits remains almost the same. On the other hand, a crawler views a lot of distant profiles and gets fewer views. Eventually it lacks credit liquidity to view the profiles of others. As such, the credit network poses a strict rate limit on profile views of the crawlers.

Genie might see a large number of honest users' activities (profile viewing) flagged due to the existence of outliers in a social network. This might limit the usability of social networks, because without

viewing a profile an outlier will not be able to befriend others. Genie also might require a fast computation of shortest paths, as for each profile viewing request, it computes all the shortest paths from viewer to viewee. Intuitively, this operation is too costly in a modern social network (more than one billion users), even considering the state of the art shortest path algorithms.

Both SpikeStrip and Genie limit crawlers' ability to quickly aggregate a significant portion of OSNs user data. Unfortunately, equipped with a large number of user profiles (fake or compromised) and employing dedicated crawlers for a long time, attackers could still collect a huge amount of users' social data.

9. MITIGATING SOCIAL SPAM

Spam is a news in web-based systems (e.g., [Xie et al., 2006; Ntoulas et al., 2006; Mehta et al., 2008]). However, OSNs have added a new flavor to it by acting as effective tools for spamming activities and propagation. Social spam (e.g., [Zinman and Donath, 2007], [Lin et al., 2007]) is unwanted content that is directed specifically at users of the OSN. The worst consequences of social spam include phishing attacks [Jagatic et al., 2007] and malware propagation [Boyd and Heer, 2006].

Spamming activity is pervasive in OSNs and spammers are successful. For example, about 0.13% of spam tweets in Twitter generate a page visit [Grier et al., 2010], which is only 0.003%-0.006% for spam email [Kanich et al., 2008]. This high click-through is due to the fact that OSNs expose intrinsic trust relationship among online friends. As such, users read and click messages or links that are shared from their friends. Study [Bilge et al., 2009] shows that 45% of users on OSNs click on links posted by their friends' accounts.

Defending spam in OSNs can improve user experience. OSN service providers will also be benefited as this will lessen the system workload in terms of dealing with unwanted communications and contents. Defense mechanisms against spam in OSNs can be classified into two categories: 1) spam content and profile detection, and 2) spam campaign detection. Spam content and profile-level detection involve checking individual accounts or contents for an evidence of spam contents (Section 9.1). On the other hand, a spam "campaign" is a collection of malicious content having a common goal, for example, selling backdoor products [Gao et al., 2010] (Section 9.2).

9.1 Spam Content and Profile Detection

Some early spam profile detections [Spitzner, 2002; Webb et al., 2008; Lee et al., 2010] use social honeypots. A honeypot is a trap deployed to capture examples of nefarious activities in networked systems [Spitzner, 2002]. For years, researchers have used honeypots to characterize malicious hacker activities [Spitzner, 2003], to obtain footprints of email address crawlers [Prince et al., 2005], and to create intrusion detection signatures [Kreibich and Crowcroft, 2004]. Social honeypots are used to monitor spammers' behaviors and store their information from the OSNs [Lee et al., 2010].

Webb et al. [Webb et al., 2008] take the first step to characterize spam in OSNs using social honeypots. They created 51 honeypot MySpace profiles in different geographic locations for harvesting deceptive spam profiles on MySpace. An automated program (commonly known as bots) works on behalf of a honeypot profile and collects all of the traffic it receives (via friend requests). After four months of the deployment and operation, the bots collected 1,570 friend requests (and corresponding spam profiles). Through statistical analysis the authors show the followings: (i) spam profiles follow distinct temporal patterns in spamming activity; (ii) 57.2% of the "About me" contents of the spam profiles are duplicated; (iii) spam profiles redirect users to predefined web pages.

In [Lee et al., 2010], the authors also collected spam profiles using social honeypots. But this work is different from the previous one in that it not only collects and characterizes spam profiles, it extracts features from the gathered spam profiles and builds classifiers to detect potential spam profiles. The

authors consider four categories of features such as demographics, content, activity and connections from the spam profiles collected from MySpace and Twitter. The paper shows the performance results of ten classifiers using the features. For MySpace dataset, each classifier's accuracy is greater than 98.4%, for Twitter dataset, each of the top 10 classifiers achieves an accuracy greater than 82.7%.

One of the limitations of these honeypot-based solutions [Webb et al., 2008; Lee et al., 2010] is that they consider all profiles that sent friend requests to honeypots are spam profiles. But in social networks, it is common to receive friend requests from unknown person, who might be legitimate users in the network. The solutions would be more rigorous if legitimate users were not considered. Also, the methods are effective when spammers become friends with the honeypots. Otherwise the honeypots will be able to target only a small subset of the spammers. Moreover, in social networks, friendship is not always required for spamming. For example, in twitter, a spammer can use mention (e.g., @user) tag to send spam tweets to a user.

Stringhini et al.'s solution [Stringhini et al., 2010] overcome some limitations of the previous two honeypot-based papers. The authors deployed honeypots accounts on Facebook, Twitter and MySpace; 300 on each platform for about one year and logged the traffic (e.g., friend requests, messages, and invitations). Combining all, the honeypots received 4,250 friend requests and 85,569 messages from the three platforms. They build classifiers from the following six features: (i) FF ratio: the ratio of the number of friend requests sent by a user and the number of friends she has; (ii) URL ratio: the ratio of the number of messages containing URLs and total messages; (iii) Message Similarity: similarity among the messages sent by a user; (iv) Friend Choice: the ratio of the total number of names among the profiles' friends, and the number of distinct first names; (v) Messages Sent: the number of messages sent by a profile as a feature; and (vi) Friend Number: the number of friends a profile has.

The authors manually inspected and labeled profiles as spam and used Random Forest algorithm for classification. A 10- fold cross validation on the training data set of Facebook yielded an estimated false positive ratio of 2% and a false negative ratio of 1%. Twitter dataset yielded an estimated false positive ratio of 2.5% and a false negative ratio of 3%. They detected 15,857 spam profiles on Twitter using the classifier and the Twitter spam team eventually suspended those accounts.

Benevenuto et al. [Benevenuto et al., 2009b; Benevenuto et al., 2009a] detect video polluters such as spammers and promoters in YouTube online video social networks using machine learning techniques. The authors considered three attribute sets: user attributes, video attributes, and social network (SN) attributes in classification. Four volunteers manually analyzed the videos and built a test set of the dataset labeling users as spammers, promoters and legitimate users. They proposed a flat classification approach, which was able to detect correctly 96% of the promoters, 57% of spammers, and wrongly classifying only 5% of the legitimate users. Interestingly, social network attributes performed the worst in classification—only one feature (UserRank) was within the top 30 features.

9.2 Spam Campaigns Detection

Chu et al. [Chu et al., 2012] detect social spam campaigns on Twitter using tweet URLs. They collected a dataset of 50 million tweets from 22 million users. They considered tweets having the same URL as a campaign and clustered the dataset into a number of campaigns. The ground truth was produced through manual inspection using Twitter's spam rules and automated URL checking in five services. They obtained a variety of features ranging from individual tweet/account levels to a collective campaign level and built a classification model to detect spam campaigns. Using several classification algorithms they were able to detect spam campaigns with more than 80% success rate. The focus of this solution is spam tweets with URLs. However, Twitter spammers can post tweets without any URL. Even obfuscated URLs (e.g., somethingDOTcom) will make the detection inefficient.

Gao et al. [Gao et al., 2010] conduct a rigorous and extensive study on detecting spam campaigns in Facebook wall posts. They crawled 187 million Facebook wall posts from about 3.5 million users. Inspired by a study [Kreibich et al., 2009] which shows that spamming bot-nets create email spam messages using templates, they consider wall posts having similar texts as a spam campaign. In particular, they model the wall posts as a graph: a post is a node and two nodes are connected by an edge if they have the same destination URL or their texts are very similar. As such, posts from the same spam campaign will make connected subgraphs or clusters. To detect which clusters are from spammers, they use "distribute" coverage and "bursty" natures of spam campaigns. The "distributed" property is characterized based on the number of user accounts posting in the cluster under the intuition that spammers will use a significant number of registered accounts for a campaign. The intuition behind the "bursty" property is that most spam campaigns are the results of coordinated actions of many accounts within short periods of time [Xie et al., 2008]. Using threshold filters on these two properties they found 297 clusters of wall posts and classified them as potentially malicious spam campaigns.

10. MITIGATING DISTRIBUTED DENIAL-OF-SERVICE ATTACKS (DDOS) ATTACKS

A denial-of-service (DOS) attack is characterized by an explicit attempt to monopolize a computer resource, so that an intended user cannot use the resource [Mirkovic et al., 2004]. A Distributed Denial-of-Service attack (DDoS) deploys multiple attacking entities to simultaneously launch the attack (we refer readers [Mirkovic and Reiher, 2004] for a taxonomy of web-based DDoS attacks and defenses). DDoS attacks in social networks are also common. For example, on August 6, 2009, Twitter, Facebook, LiveJournal, Google's Blogger, and YouTube were attacked by a DDoS attack [McCarthy, 2009]. Twitter experienced interrupted service for several hours, users were complaining of not being able to send their Tweets. Facebook users were experiencing longer periods of time (delays) in loading Facebook pages.

Several papers evaluated how a social network could be leveraged to launch a bot-net based DDoS on any target of the internet, including the social network itself. Athanasopoulos et al. [Athanasopoulos et al., 2008] introduce a bot-net "FaceBot" that uses a social network to carry out a DDoS attack against any host on the internet (including the social network itself). They created a real-world Facebook application, "Photo of the Day", that presents a different photo from National Geographic to Facebook users every day. Every time a user clicks on the application, an image from the National Geographic appears. However, they placed special codes in the application's source code. Every time a user views the photo, this code sends a HTTP request towards a victim host, which causes the victim to serve a request of 600 KBytes. They used a web server as a victim and observed that the server recorded 6 Mbit per second of traffic. They introduce defense mechanisms which include providing application developers with a strict API that is capable of giving access to resources only related to the system.

Ur and Ganapathy [Ur and Ganapathy, 2009] showed how malicious social network users can leverage their connections with hubs to launch DDoS attacks. They created MySpace profiles which befriended hubs in the network. Those profiles posted "hotlinks" to large media files hosted by a victim web server to Hubs' pages. As hubs receive a large number of hits, a significant number of the visitors would click those hotlinks. As a consequence, it staged a scenario where a flash crowd was sending requests to the victim web server—a denial of service was the result. They proposed several mitigating techniques. One approach is to restrict some privileges of a user when he becomes a hub (e.g., friends of a hub might no longer be able to post comments containing HTML tags to the hub's page). But this approach unfortunately restricts the user's freedom on the OSN. So, they propose a focused automated monitoring on a hub or creating a hierarchy of a hub's friend, so that only close friends will be able to post on a Hub's profile (the intuition is that close friends will not exploit the hub). Furthermore, they

recommend a reputation based system for social networks that scores user behavior. Only users with a higher reputation scores are allowed to post on the Hub's profile.

However, bot-net based DDoS attacks are difficult to mitigate, because of the difficulty to distinguish legitimate communications from those that are part of the attack. As social networks are flourishing, bot-net based DDoS attacks are becoming stronger, because more legitimate users are unwillingly becoming part of an attack.

11. MITIGATING MALWARE ATTACKS

Malicious software (malware) is a program that is specifically designed to gain access, disrupt computer operation, gather sensitive information or damage a computer without the knowledge of the owner. Participatory internet technologies (e.g., AJAX) and applications (e.g., RSS) have expedited malware attacks, because they enable the participation of the users. OSNs (all of them use participatory technologies and applications) are providing themselves as infrastructures for propagating malware. The "Koobface" is probably the best example of malware propagation using social networks [Facebook, 2012]. It spread rapidly through Facebook social networks. The malware used Facebook credentials on a compromised computer and sent messages to the owner's Facebook friends. The messages redirected the owner's friends to a third-party website and they were asked to download an update of the Adobe Flash player. If they would download and install the file, Koobface would install and infect their system using the same process.

In a survey, Gao et al. [Gao et al., 2011] discuss a number of methods in which malware propagates through social networks. For example, using cross-site request forgery (CSRF or XSRF) malware invites legitimate users to click on a link. If a user clicks, it opens an exploited page containing malicious scripts. Eventually, the malware submits a message with a URL for a wall post on the user's profile and clicks on the "Share" button so that all of her friends can see this message as well as the link. URL obfuscations are also widely used for malware attacks. An attacker uses commonly known URL shorteners to obfuscate the true location of a link and lures other users to click it.

Unfortunately, malware propagation on social networks exhibits unique propagation vectors. As such, existing Internet worm detection techniques (e.g., [Ellis et al., 2004]) cannot be applied to them. In the context of OSNs, Xu et al. [Xu et al., 2010b] proposed an OSN malware detection system by leveraging both the propagation characteristics of the malware and the topological properties of OSNs. They introduced a "maximum coverage algorithm" that picks a subset of legitimate OSN users to whom the defense system attaches "decoy friends" to monitor the entire social graph. When the decoy friends receive suspicious malware propagation evidence, the detection system performs local and network correlations to distinguish actual malware evidence from normal user communication. However, the challenge for this honeypot-based approach is to determine how many social honeypots (in this context decoy friends) large-scale OSNs (e.g., billions of Facebook users) should deploy.

12. SUMMARY AND DISCUSSION

Millions of Internet users are using OSNs for communication and collaboration. Many companies rely on OSNs for promoting their products and influencing the market. It becomes harder and harder to imagine life without the use of OSN tools, whether for creating an image of oneself or organization, for selectively following news as filtered by the group of friends, or for keeping in touch. However, the growing reliance on OSNs is impaired by an increasingly more sophisticated range of attacks that undermine the very usefulness of the OSNs.

This paper reviews online social networks' privacy and security issues. We have categorized various attacks on OSNs based on social network stakeholders and the forms of attack targeted at them. Specifically, we have categorized those attacks as attacks on users and attacks on the OSN. We have

discussed how the attacks are launched, what are the available defense techniques and what are the challenges involved in such defenses.

In online social networks, privacy and security issues are not separable. In some contexts privacy and security goals may be the same, but there are other contexts where they may be orthogonal, and there are also contexts where they are in conflict. For example, in an OSN, a user wants privacy when she is communicating with other users though the messaging service. She will expect that non-recipients of the message will not be able to read it. OSN services will ensure this by providing a secure communication channel. In this context, the goals of security and privacy are the same. Consider another context where there is a security goal of authenticating a user's account. OSNs usually do this by sending an activation link as a message to the user's e-mail address. This is not a privacy issue—OSNs are just securely authenticating that malicious users are not using the legitimate user's e-mail to register. In this context, security and privacy goals are orthogonal. However, anonymous views in OSNs (e.g., LinkedIn) present a context where security and privacy goals are in conflict. Users may want to have privacy (e.g., anonymization) while viewing other users' profiles. However, the viewee might want to secure her profile from anonymous viewing.

The attacks discussed in this paper are often closely intertwined. User data collected though crawling attacks or via social applications may help an attacker to create background knowledge for launching de-anonymization attacks. An attacker might possess an unprecedented number of user accounts using malware and Sybil attacks and could use those accounts for social spam propagation and distributed denial-of-service attacks. Social spam can also be used to propagate malware. Some attacks might be a pre-requisite for another attacks. For example, a de-anonymization attack can reveal the identity of an individual. That identity could be used to launch an inference attack and to learn unspecified attributes of an individual.

There are also several functionality-oriented attacks that we did not discuss in this paper. Functionality-oriented attacks attempt to exploit specific functionalities of a social network. For example, Location-based Services (LSP) such as Foursquare, Loopt and Facebook Places utilize geo-location information to publish users' checked-in places. In some LSP, users can accumulate "points" for "checking in" at certain venues or locations and can get real-world discounts or freebies in exchange for these points. There is a body of research that analyzes the technical feasibility of anonymous usage of location-based services so that users are not impacted by location sharing [Gruteser and Grunwald, 2003; Xiao and Tao, 2006]. Moreover, real-world rewards and discounts give incentives for users in LSP to cheat on their locations, and hence research [Zhu and Cao, 2011; He et al., 2011] has focused on how to prevent users from location cheating.

OSNs and social applications are here to stay, and while they mature, new security and privacy attacks will take shape. Technical advances in this area can only be of limited effect if not supported by legislative measures for protecting the user from other users and from the service providers [Nissenbaum, 2011].

REFERENCES

Acohido, B. (2009). Cybergangs use cheap labor to break codes on social sites. http://usatoday30.usatoday.com/tech/news/computersecurity/2009-04-22-captcha-code-breakers_N.htm/.

Acquisti, A. and Gross, R. (2006). Imagined communities: Awareness, information sharing, and privacy on the facebook. In *Privacy enhancing technologies*, pages 36–58. Springer.

Adar, E., Weld, D. S., Bershad, B. N., and Gribble, S. S. (2007). Why we search: visualizing and predicting user behavior. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 161–170, New York, NY, USA. ACM.

Adu-Oppong, F., Gardiner, C. K., Kapadia, A., and Tsang, P. P. (2008). Social circles: Tackling privacy in social networks. In *Symposium on Usable Privacy and Security (SOUPS)*.

Aiello, L., Milanesio, M., Ruffo, G., and Schifanella, R. (2008). Tempering kademlia with a robust identity based system. In *Peer-to-Peer Computing , 2008. P2P '08. Eighth International Conference on*, pages 30–39.

Aiello, L. M. and Ruffo, G. (2012). Lotusnet: tunable privacy for distributed online social network services. *Computer Communications*, 35(1):75–88.

Alexa (2014). The top 500 sites on the web. http://www.alexa.com/topsites.

Arrington, M. (2006). Aol proudly releases massive amounts of private data. http://techcrunch.com/2006/08/06/aol-proudly-releases-massive-amounts-of-user-search-data/.

Athanasopoulos, E., Makridakis, A., Antonatos, S., Antoniades, D., Ioannidis, S., Anagnostakis, K. G., and Markatos, E. P. (2008). Antisocial networks: Turning a social network into a botnet. In *11th International Conference on Information Security*, pages 146–160. Springer.

Backstrom, L., Dwork, C., and Kleinberg, J. (2007). Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 181–190, New York, NY, USA. ACM.

Baden, R., Bender, A., Spring, N., Bhattacharjee, B., and Starin, D. (2009). Persona: an online social network with user-defined privacy. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, SIGCOMM '09, pages 135–146, New York, NY, USA. ACM.

Banks, L. and Wu, S. (2009). All friends are not created equal: An interaction intensity based approach to privacy in online social networks. In *Computational Science and Engineering, 2009. CSE '09. International Conference on*, volume 4, pages 970–974.

BBC (2012). Facebook has more than 83 million illegitimate accounts. http://www.bbc.co.uk/news/technology-19093078.

Benevenuto, F., Rodrigues, T., Almeida, J., Goncalves, M., and Almeida, V. (2009a). Detecting spammers and content promoters in online video social networks. In *INFOCOM Workshops 2009, IEEE*, pages 1–2.

Benevenuto, F., Rodrigues, T., Almeida, V., Almeida, J., and Gonçalves, M. (2009b). Detecting spammers and content promoters in online video social networks. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 620–627, New York, NY, USA. ACM.

Besmer, A., Lipford, H. R., Shehab, M., and Cheek, G. (2009). Social applications: exploring a more secure framework. In *Proceedings of the 5th Symposium on Usable Privacy and Security*, SOUPS '09, pages 2:1–2:10, New York, NY, USA. ACM.

Bilge, L., Strufe, T., Balzarotti, D., and Kirda, E. (2009). All your contacts are belong to us: automated identity theft attacks on social networks. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 551–560, New York, NY, USA. ACM.

Bonneau, J., Anderson, J., and Danezis, G. (2009). Prying data out of a social network. In *Social Network Analysis and Mining, 2009. ASONAM '09. International Conference on Advances in*, pages 249–254.

Bonneau, J. and Preibusch, S. (2010). The privacy jungle: On the market for data protection in social networks. In *Economics of information security and privacy*, pages 121–167. Springer.

Boshmaf, Y., Muslukhov, I., Beznosov, K., and Ripeanu, M. (2011). The socialbot network: when bots socialize for fame and money. In *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, pages 93–102, New York, NY, USA. ACM.

Boyd, D. (2004). Friendster and publiclly articulated social networking. In *Extended Abstracts of the Conference on Human Factors and Computing Systems (CHI 2004)*, pages 1279–1282.

Boyd, D. and Heer, J. (2006). Profiles as conversation: Networked identity performance on friendster. In *System Sciences, 2006. HICSS '06. Proceedings of the 39th Annual Hawaii International Conference on*, volume 3, pages 59c–59c.

Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. (2005). Gossip algorithms: design, analysis and applications. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1653 – 1664 vol. 3.

Bradley, T. (2012). 45,000 facebook accounts compromised: What to know. http://bit.ly/TUY3i8.

Brandes, U. (2001). A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 40(2):163–177.

Buchegger, S., Schiöberg, D., Vu, L.-H., and Datta, A. (2009). Peerson: P2p social networking: early experiences and insights. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, SNS '09, pages 46–52, New York, NY, USA. ACM.

Campan, A. and Truta, T. M. (2009). Data and structural k-anonymity in social networks. In *Privacy, Security, and Trust in KDD*, pages 33–54. Springer.

Cao, Q., Sirivianos, M., Yang, X., and Pregueiro, T. (2012). Aiding the detection of fake accounts in large scale social online services. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, NSDI'12, pages 15–15, Berkeley, CA, USA. USENIX Association.

Carminati, B., Ferrari, E., Heatherly, R., Kantarcioglu, M., and Thuraisingham, B. (2009). A semantic web based framework for social network access control. In *Proceedings of the 14th ACM symposium on Access control models and technologies*, SACMAT '09.

Carminati, B., Ferrari, E., and Perego, A. (2006). Rule-based access control for social networks. In *Proceedings of the 2006 international conference on On the Move to Meaningful Internet Systems*, pages 1734–1744.

Cellan-Jones, R. (2012). Facebook 'likes' and adverts' value doubted. http://www.bbc.co.uk/news/technology-18813237.

Chiluka, N., Andrade, N., Pouwelse, J., and Sips, H. (2012). Leveraging trust and distrust for sybil-tolerant voting in online social media. In *Proceedings of the 1st Workshop on Privacy and Security in Online Social Media*, PSOSM '12, pages 1:1–1:8, New York, NY, USA. ACM.

Choi, H. C., Kruk, S. R., Grzonkowski, S., Stankiewicz, K., Davis, B., and Breslin, J. (2006). Trust models for community aware identity management. In *Proceedings of the Identity, Reference and Web Workshop, in conjunction with WWW 2006*, page 140154.

Chu, Z., Widjaja, I., and Wang, H. (2012). Detecting social spam campaigns on twitter. In *In Proceedings of Conference on Applied Cryptography and Network Security*, Lecture Notes in Computer Science, pages 455–472. Springer Berlin Heidelberg.

Conti, M., Hasani, A., and Crispo, B. (2011). Virtual private social networks. In *Proceedings of the first ACM conference on Data and application security and privacy*, CODASPY '11, pages 39–50, New York, NY, USA. ACM.

Cummings, J. N., Butler, B., and Kraut, R. (2002). The quality of online social relationships. *Commun. ACM*, 45(7):103–108.

Cutillo, L., Molva, R., and Strufe, T. (2009a). Privacy preserving social networking through decentralization. In *Wireless On-Demand Network Systems and Services, 2009. WONS 2009. Sixth International Conference on*, pages 145–152.

Cutillo, L., Molva, R., and Strufe, T. (2009b). Safebook: A privacy-preserving online social network leveraging on real-life trust. *Communications Magazine, IEEE*, 47(12):94–101.

Dam, W. B. (2009). School teacher suspended for facebook gun photo. http://www.foxnews.com/story/2009/02/05/schoolteacher-suspended-for-facebook-gun-photo/.

Dandekar, P., Goel, A., Wellman, M. P., and Wiedenbeck, B. (2012). Strategic formation of credit networks. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 559–568, New York, NY, USA. ACM.

Danezis, G. (2009). Inferring privacy policies for social networking services. In *Proceedings of the 2nd ACM workshop on Security and artificial intelligence*, AISec '09, pages 5–10, New York, NY, USA. ACM.

Danezis, G. and Mittal., P. (2009). Sybilinfer: Detecting sybil nodes using social networks. In *Network and Distributed System Security Symposium (NDSS)*.

DeFigueiredo, D. and Barr, E. (July). Trustdavis: a non-exploitable online reputation system. In *E-Commerce Technology, 2005. CEC 2005. Seventh IEEE International Conference on*, pages 274–283.

Devriese, D. and Piessens, F. (2010). Noninterference through secure multi-execution. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pages 109–124, Washington, DC, USA. IEEE Computer Society.

Dey, R., Tang, C., Ross, K., and Saxena, N. (2012). Estimating age privacy leakage in online social networks. In *INFOCOM, 2012 Proceedings IEEE*, pages 2836–2840.

Douceur, J. (2002). The sybil attack. In Druschel, P., Kaashoek, F., and Rowstron, A., editors, *Peer-to-Peer Systems*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer Berlin Heidelberg.

Dwyer, C. (2011). Privacy in the age of google and facebook. *Technology and Society Magazine, IEEE*, 30(3):58–63.

Egele, M., Moser, A., Kruegel, C., and Kirda, E. (2012). Pox: Protecting users from malicious facebook applications. *Computer Communications*, 35(12).

Elahi, N., Chowdhury, M., and Noll, J. (2008). Semantic access control in web based communities. In *Proceedings of the 2008 The Third International Multi-Conference on Computing in the GlobalInformation Technology*, pages 131 –136.

Ellis, D. R., Aiken, J. G., Attwood, K. S., and Tenaglia, S. D. (2004). A behavioral approach to worm detection. In *Proceedings of the 2004 ACM workshop on Rapid malcode*, WORM '04, pages 43–53, New York, NY, USA. ACM.

Engelmore, R., editor (1988). *Readings from the AI magazine*. American Association for Artificial Intelligence, Menlo Park, CA, USA.

Facebook (2012). Facebook's continued fight against koobface. http://on.fb.me/y5ibe1.

Facebook (2015). Statement of rights and responsibilities. https://www.facebook.com/legal/terms.

Fang, L. and LeFevre, K. (2010). Privacy wizards for social networking sites. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 351–360, New York, NY, USA. ACM.

Felt, A. and Evans, D. (2008). Privacy protection for social networking apis. In *2008 Web 2.0 Security and Privacy (W2SP08)*.

Fiesler, C. and Bruckman, A. (2014). Copyright terms in online creative communities. In *CHI'14 Extended Abstracts on Human Factors in Computing Systems*, pages 2551–2556. ACM.

Flaxman, A. (2008). Expansion and lack thereof in randomly perturbed graphs. *Algorithms and Models for the Web-Graph*, 4936:24–35.

Fong, P. (2011a). Preventing sybil attacks by privilege attenuation: A design principle for social network systems. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 263–278.

Fong, P. W. (2011b). Relationship-based access control: protection model and policy language. In *Proceedings of the first ACM conference on Data and application security and privacy*, pages 191–202.

Gao, H., Hu, J., Huang, T., Wang, J., and Chen, Y. (2011). Security issues in online social networks. *Internet Computing, IEEE*, 15(4):56–63.

Gao, H., Hu, J., Wilson, C., Li, Z., Chen, Y., and Zhao, B. Y. (2010). Detecting and characterizing social spam campaigns. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, IMC '10, pages 35–47, New York, NY, USA. ACM.

Ghosh, A., Mahdian, M., Reeves, D., Pennock, D., and Fugger, R. (2007). Mechanism design on trust networks. In Deng, X. and Graham, F., editors, *Internet and Network Economics*, volume 4858 of *Lecture Notes in Computer Science*, pages 257–268. Springer Berlin Heidelberg.

Giunchiglia, F., Zhang, R., and Crispo, B. (2008). Relbac: Relation based access control. In *Fourth International Conference on Semantics, Knowledge and Grid*, pages 3 –11.

Graffi, K., Gross, C., Stingl, D., Hartung, D., Kovacevic, A., and Steinmetz, R. (2011). Lifesocial.kom: A secure and p2p-based solution for online social networks. In *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, pages 554–558.

Grier, C., Thomas, K., Paxson, V., and Zhang, M. (2010). @spam: the underground on 140 characters or less. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 27–37, New York, NY, USA. ACM.

Gross, R. and Acquisti, A. (2005a). Information revelation and privacy in online social networks. In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, WPES '05, pages 71–80, New York, NY, USA. ACM.

Gross, R. and Acquisti, A. (2005b). Information revelation and privacy in online social networks. In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80.

Gruteser, M. and Grunwald, D. (2003). Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, MobiSys '03, pages 31–42, New York, NY, USA. ACM.

Gubichev, A., Bedathur, S., Seufert, S., and Weikum, G. (2010). Fast and accurate estimation of shortest paths in large graphs. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 499–508, New York, NY, USA. ACM.

Guha, S., Tang, K., and Francis, P. (2008). Noyb: privacy in online social networks. In *Proceedings of the first workshop on Online social networks*, WOSN '08, pages 49–54, New York, NY, USA. ACM.

Hay, M., Miklau, G., Jensen, D., Towsley, D., and Weis, P. (2008). Resisting structural re-identification in anonymized social networks. *Proc. VLDB Endow.*, 1(1):102–114.

He, W., Liu, X., and Ren, M. (2011). Location cheating: A security challenge to location-based social network services. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 740–749. IEEE.

Heatherly, R., Kantarcioglu, M., and Thuraisingham, B. (2013). Preventing private information inference attacks on social networks. *Knowledge and Data Engineering, IEEE Transactions on*, 25(8):1849–1862.

Heymann, P., Koutrika, G., and Garcia-Molina, H. (2007). Fighting spam on social web sites: A survey of approaches and future challenges. *IEEE Internet Computing*, 11(6):36–45.

Holt, R. (2013). Twitter in numbers. http://www.telegraph.co.uk/technology/twitter/9945505/Twitter-in-numbers.html.

Hwang, T., Pearce, I., and Nanis, M. (2012). Socialbots: Voices from the fronts. *interactions*, 19(2):38–45.

Irani, D., Balduzzi, M., Balzarotti, D., Kirda, E., and Pu, C. (2011). Reverse social engineering attacks in online social networks. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 55–74. Springer.

Irani, D., Webb, S., and Pu, C. (2010). Study of static classification of social spam profiles in myspace. In *Proceedings of the 4th International Conference on Weblogs and Social Media*.

Jagatic, T. N., Johnson, N. A., Jakobsson, M., and Menczer, F. (2007). Social phishing. *Commun. ACM*, 50(10):94–100.

Jansen, B. J. and Booth, D. (2010). Classifying web queries by topic and user intent. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, pages 4285–4290, New York, NY, USA. ACM.

Jurek, M. (2011). Google explores +1 button to influence search results. http://www.tekgoblin.com/2011/08/29/google-explores-1-button-to-influence-search-results/.

Kanich, C., Kreibich, C., Levchenko, K., Enright, B., Voelker, G. M., Paxson, V., and Savage, S. (2008). Spamalytics: an empirical analysis of spam marketing conversion. In *Proceedings of the 15th ACM conference on Computer and communications security*, CCS '08, pages 3–14, New York, NY, USA. ACM.

Kayes, I. and Iamnitchi, A. (2013a). Aegis: A semantic implementation of privacy as contextual integrity in social ecosystems. In *11th International Conference on Privacy, Security and Trust (PST)*.

Kayes, I. and Iamnitchi, A. (2013b). Out of the wild: On generating default policies in social ecosystems. In *IEEE ICC'13 - Workshop on Beyond Social Networks: Collective Awareness*.

Kelly, S. (2008). Identity 'at risk on facebook. http://news.bbc.co.uk/2/hi/programmes/click_online/7375772.stm.

Klimt, B. and Yang, Y. (2004). The enron corpus: A new dataset for email classification research. In *15th European Conference on Machine Learning*, pages 217–226. Springer.

Kourtellis, N., Finnis, J., Anderson, P., Blackburn, J., Borcea, C., and Iamnitchi, A. (2010). Prometheus: User-controlled p2p social data management for socially-aware applications. In *11th International Middleware Conference*.

Kreibich, C. and Crowcroft, J. (2004). Honeycomb: creating intrusion detection signatures using honeypots. *SIGCOMM Comput. Commun. Rev.*, 34(1):51–56.

Kreibich, C., Kanich, C., Levchenko, K., Enright, B., Voelker, G. M., Paxson, V., and Savage, S. (2009). Spamcraft: An inside look at spam campaign orchestration. *Proc. of 2nd USENIX LEET*.

Krishnamurthy, B., Gill, P., and Arlitt, M. (2008). A few chirps about twitter. In *Proceedings of the first workshop on Online social networks*, pages 19–24.

Krishnamurthy, B. and Wills, C. E. (2008). Characterizing privacy in online social networks. In *Proceedings of the first workshop on Online social networks*, pages 37–42.

Kruk, S. (2004). Foaf-realm: control your friends access to the resource. In *In Proceedings of the 1st Workshop on Friend of a Friend*.

Kwak, H., Lee, C., Park, H., and Moon, S. (2010). What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 591–600, New York, NY, USA. ACM.

Langville, A. N. and Meyer, C. D. (2004). Deeper inside pagerank. *Internet Mathematics*, 1:2004.

Lee, K., Caverlee, J., and Webb, S. (2010). Uncovering social spammers: social honeypots + machine learning. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 435–442, New York, NY, USA. ACM.

Lenhart, A., Purcell, K., Smith, A., and Zickuhr, K. (2010). Social media and young adults. http://bit.ly/cQdgi3.

Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, pages 3–12, New York, NY, USA. Springer-Verlag New York, Inc.

Lin, Y.-R., Sundaram, H., Chi, Y., Tatemura, J., and Tseng, B. L. (2007). Splog detection using self-similarity analysis on blog temporal dynamics. In *Proceedings of the 3rd international workshop on Adversarial information retrieval on the web*, AIRWeb '07, pages 1–8, New York, NY, USA. ACM.

Lindamood, J., Heatherly, R., Kantarcioglu, M., and Thuraisingham, B. (2009). Inferring private information using social network data. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 1145–1146, New York, NY, USA. ACM.

Lipford, H. R., Besmer, A., and Watson, J. (2008). Understanding privacy settings in facebook with an audience view. *UPSEC*, 8:1–8.

Liu, K. and Terzi, E. (2008). Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 93–106, New York, NY, USA. ACM.

Liu, Y., Gummadi, K. P., Krishnamurthy, B., and Mislove, A. (2011). Analyzing facebook privacy settings: user expectations vs. reality. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 61–70.

Lotan, G., Graeff, E., Ananny, M., Gaffney, D., Pearce, I., et al. (2011). The arab spring— the revolutions were tweeted: Information flows during the 2011 tunisian and egyptian revolutions. *International Journal of Communication*, 5:31.

Lucas, M. M. and Borisov, N. (2008). Flybynight: mitigating the privacy risks of social networking. In *Proceedings of the 7th ACM workshop on Privacy in the electronic society*, WPES '08, pages 1–8, New York, NY, USA. ACM.

Luo, W., Xie, Q., and Hengartner, U. (2009). Facecloak: An architecture for user privacy on social networking sites. In *Computational Science and Engineering, 2009. CSE '09. International Conference on*, volume 3, pages 26–33.

Madejski, M., Johnson, M. L., and Bellovin, S. M. (2011). The failure of online social network privacy settings. *Department of Computer Science, Columbia University*.

Mail, D. (2011). Bank worker fired for facebook post comparing her 7-an-hour wage to lloyds boss's $4,000$-an-hour salary. http://dailym.ai/fjRTlC.

Marinando (2010). Tuenti, spain's leading social network, switches on local for a location-based future. http://techcrunch.com/2010/03/25/tuenti-spains_leading_social_network_mixes_local_into_social_in_a_very_big_way/.

Masoumzadeh, A. and Joshi, J. (2011). Ontology-based access control for social network systems. *IJIPSI*, 1(1):59–78.

McCarthy, C. (2009). Twitter crippled by denial-of-service attack. http://news.cnet.com/8301-13577_3-10304633-36.html.

Mehta, B., Nangia, S., Gupta, M., and Nejdl, W. (2008). Detecting image spam using visual features and near duplicate detection. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 497–506, New York, NY, USA. ACM.

Mills, E. (2008). Facebook suspends app that permitted peephole. http://news.cnet.com/8301-10784_3-9977762-7.html.

Mirkovic, J., Dietrich, S., Dittrich, D., and Reiher, P. (2004). *Internet Denial of Service: Attack and Defense Mechanisms (Radia Perlman Computer Networking and Security)*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Mirkovic, J. and Reiher, P. (2004). A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53.

Mishra, N., Schreiber, R., Stanton, I., and Tarjan, R. E. (2007). Clustering social networks. In *Proceedings of the 5th international conference on Algorithms and models for the web-graph*, WAW'07, pages 56–67, Berlin, Heidelberg. Springer-Verlag.

Mislove, A., Post, A., Druschel, P., and Gummadi, K. P. (2008). Ostra: leveraging trust to thwart unwanted communication. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'08, pages 15–30, Berkeley, CA, USA. USENIX Association.

Mohaisen, A., Yun, A., and Kim, Y. (2010). Measuring the mixing time of social graphs. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, IMC '10, pages 383–389, New York, NY, USA. ACM.

Mondal, M., Viswanath, B., Clement, A., Druschel, P., Gummadi, K. P., Mislove, A., and Post, A. (2012). Defending against large-scale crawls in online social networks. In *Proceedings of the 8th ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT'12)*, Nice, France.

Motoyama, M., McCoy, D., Levchenko, K., Savage, S., and Voelker, G. M. (2011). Dirty jobs: the role of freelance labor in web service abuse. In *Proceedings of the 20th USENIX conference on Security*, SEC'11, pages 14–14, Berkeley, CA, USA. USENIX Association.

Murphy, S. (2010). Teens ditch e-mail for texting and facebook. http://www.nbcnews.com/id/38585236/.

Narayanan, A., Shi, E., and Rubinstein, B. I. (2011). Link prediction by de-anonymization: How we won the kaggle social network challenge. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1825–1834. IEEE.

Narayanan, A. and Shmatikov, V. (2009). De-anonymizing social networks. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 173–187.

Nazir, A., Raza, S., Chuah, C.-N., and Schipper, B. (2010). Ghostbusting facebook: detecting and characterizing phantom profiles in online social gaming applications. In *Proceedings of the 3rd conference on Online social networks*, WOSN'10, pages 1–1, Berkeley, CA, USA. USENIX Association.

Newman, M. E. J. (2010). *Networks: An Introduction*. Oxford University Press.

Nielsen (2012). Social networks & blogs now 4^{th} most popular online activity, ahead of personal e-mail. http://www.nielsen.com/us/en/press-room/2009/social_networks__.html.

Nissenbaum, H. (2004). Privacy as contextual integrity. *Washington Law Review*, 79(1):119–158.

Nissenbaum, H. (2011). A contextual approach to privacy online. *Daedalus*, 140(4):32–48.

Ntoulas, A., Najork, M., Manasse, M., and Fetterly, D. (2006). Detecting spam web pages through content analysis. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 83–92, New York, NY, USA. ACM.

Nunes, S., Ribeiro, C., and David, G. (2008). Use of temporal expressions in web search. In *Proceedings of the IR research, 30th European conference on Advances in information retrieval*, ECIR'08, pages 580–584, Berlin, Heidelberg. Springer-Verlag.

Paul, T., Stopczynski, M., Puscher, D., Volkamer, M., and Strufe, T. (2012). C4ps - helping facebookers manage their privacy settings. In *Social Informatics*, pages 188–201.

Peterson, R. S. and Sirer, E. G. (2009). Antfarm: efficient content distribution with managed swarms. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, pages 107–122.

Piatek, M., Isdal, T., Krishnamurthy, A., and Anderson, T. (2008). One hop reputations for peer to peer file sharing workloads. In *NSDI'08*.

Post, A., Shah, V., and Mislove, A. (2011). Bazaar: strengthening user reputations in online marketplaces. In *Proceedings of the 8th USENIX conference on Networked systems design and implementation*, NSDI'11, pages 14–14, Berkeley, CA, USA. USENIX Association.

Prince, M., Dahl, B., Holloway, L., Keller, A., and Langheinrich, E. (2005). Understanding how spammers steal your e-mail address: An analysis of the first six months of data from project honey pot. In *Second Conference on Email and Anti-Spam*.

Quercia, D. and Hailes, S. (2010). Sybil attacks against mobile users: Friends and foes to the rescue. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–5.

Ratkiewicz, J., Conover, M., Meiss, M., Gonçalves, B., Flammini, A., and Menczer, F. (2011). Detecting and tracking political abuse in social media. In *Proc. of ICWSM*.

Reynaert, T., De Groef, W., Devriese, D., Desmet, L., and Piessens, F. (2012). Pesap: A privacy enhanced social application platform. In *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*, pages 827–833.

Riley, D. (2007). Stat gaming services come to youtube. http://www.bbc.co.uk/news/technology-18813237.

Russell, S. J., Norvig, P., Canny, J. F., Malik, J. M., and Edwards, D. D. (1995). *Artificial intelligence: a modern approach*, volume 74. Prentice hall Englewood Cliffs.

Saltzer, J. and Schroeder, M. (1975). The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308.

Shakimov, A., Lim, H., Caceres, R., Cox, L., Li, K., Liu, D., and Varshavsky, A. (2011). Vis- a-vis: Privacy-preserving online social networking via virtual individual servers. In *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*, pages 1–10.

Shehab, M., Cheek, G., Touati, H., Squicciarini, A., and Cheng, P.-C. (2010). User centric policy management in online social networks. In *IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY)*, pages 9 –13.

Shetty, J. and Adibi, J. (2005). Discovering important nodes through graph entropy the case of enron email database. In *Proceedings of the 3rd international workshop on Link discovery*, LinkKDD '05, pages 74–81, New York, NY, USA. ACM.

Simpson, A. (2008). On the need for user-defined fine-grained access control policies for social networking applications. In *Proceedings of the workshop on Security in Opportunistic and SOCial networks*, SOSOC '08, pages 1:1–1:8, New York, NY, USA. ACM.

Singh, K., Bhola, S., and Lee, W. (2009). xbook: redesigning privacy control in social networking platforms. In *Proceedings of the 18th conference on USENIX security symposium*, SSYM'09, pages 249–266, Berkeley, CA, USA. USENIX Association.

Spitzner, L. (2002). *Honeypots tracking hackers*. Addison-Wesley, 1 edition.

Spitzner, L. (2003). The honeynet project: trapping the hackers. *Security Privacy, IEEE*, 1(2):15–23.

Squicciarini, A., Paci, F., and Sundareswaran, S. (2010). Prima: an effective privacy protection mechanism for social networks. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 320–323.

Staff, E. (2010). Verisign: 1.5m facebook accounts for sale in web forum. http://www.pcmag.com/article2/0,2817,2363004,00.asp.

Steel, E. and Fowler, G. A. (2010). Facebook in privacy breach. http://online.wsj.com/article/SB10001424052702304772804575558484075236968.html.

Stein, T., Chen, E., and Mangla, K. (2011). Facebook immune system. In *Proceedings of the 4th Workshop on Social Network Systems*, SNS '11, pages 8:1–8:8, New York, NY, USA. ACM.

Strater, K. and Lipford, H. R. (2008). Strategies and struggles with privacy in an online social networking community. In *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction - Volume 1*, BCS-HCI '08, pages 111–119, Swinton, UK, UK. British Computer Society.

Stringhini, G., Kruegel, C., and Vigna, G. (2010). Detecting spammers on social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, pages 1–9, New York, NY, USA. ACM.

Stringhini, G., Wang, G., Egele, M., Kruegel, C., Vigna, G., Zheng, H., and Zhao, B. Y. (2013). Follow the green: Growth and dynamics in twitter follower markets. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 163–176, New York, NY, USA. ACM.

Strufe, T. (2010). Profile popularity in a business-oriented online social network. In *Proceedings of the 3rd Workshop on Social Network Systems*, SNS '10, pages 2:1–2:6, New York, NY, USA. ACM.

Sweeney, L. (2000). Uniqueness of simple demographics in the us population. *Carnegie Mellon University Laboratory for International Data Privacy*.

Sweeney, L. (2002). k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570.

Thomas, K., Grier, C., Song, D., and Paxson, V. (2011). Suspended accounts in retrospect: an analysis of twitter spam. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, IMC '11, pages 243–258, New York, NY, USA. ACM.

Tran, N., Min, B., Li, J., and Subramanian, L. (2009). Sybil-resilient online content voting. In *In Proceedings of the 6th Symposium on Networked System Design and Implementation (NSDI*.

Tsuchiya, P. F. (1988). The landmark hierarchy: a new hierarchy for routing in very large networks. *SIGCOMM Comput. Commun. Rev.*, 18(4):35–42.

Ur, B. E. and Ganapathy, V. (2009). Evaluating attack amplification in online social networks. In *Proceedings of the 2009 Web 2.0 Security and Privacy Workshop*. Citeseer.

Viswanath, B., Mondal, M., Clement, A., Druschel, P., Gummadi, K., Mislove, A., and Post, A. (2012a). Exploring the design space of social network-based sybil defenses. In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, pages 1–8.

Viswanath, B., Mondal, M., Gummadi, K. P., Mislove, A., and Post, A. (2012b). Canal: scaling social network-based sybil tolerance schemes. In *Proceedings of the 7th ACM european conference on Computer Systems*, EuroSys '12, pages 309–322, New York, NY, USA. ACM.

Viswanath, B., Post, A., Gummadi, K. P., and Mislove, A. (2010). An analysis of social network-based sybil defenses. In *Proceedings of the ACM SIGCOMM 2010 conference*, SIGCOMM '10, pages 363–374, New York, NY, USA. ACM.

von Ahn, L., Blum, M., and Langford, J. (2004). Telling humans and computers apart automatically. *Commun. ACM*, 47(2):56–60.

Wagner, C., Mitter, S., Körner, C., and Strohmaier, M. (2012). When social bots attack: Modeling susceptibility of users in online social networks. In *Proceedings of the WWW*, volume 12.

Walsh, K. and Sirer, E. G. (2006). Experience with an object reputation system for peer-to-peer filesharing. In *NSDI'06*.

Wang, G., Mohanlal, M., Wilson, C., Xiao Wang, M. M., Zheng, H., and Zhao, B. Y. (2013). Social turing tests: Crowdsourcing sybil detection. In *In Proceedings of The 20th Annual Network & Distributed System Security Symposium (NDSS)*.

Webb, S., Caverlee, J., and Pu, C. (2008). Social honeypots: Making friends with a spammer near you. In *Proceedings of the Fifth Conference on Email and Anti-Spam (CEAS 2008), Mountain View, CA*.

Wei, W., Xu, F., Tan, C., and Li, Q. (March). Sybildefender: Defend against sybil attacks in large social networks. In *INFOCOM, 2012 Proceedings IEEE*, pages 1951–1959.

Wilson, C., Sala, A., Bonneau, J., Zablit, R., and Zhao, B. Y. (2010). Don't tread on me: moderating access to osn data with spikestrip. In *Proceedings of the 3rd conference on Online social networks*, WOSN'10, pages 5–5, Berkeley, CA, USA. USENIX Association.

Wu, X., Ying, X., Liu, K., and Chen, L. (2010). *A survey of privacy-preservation of graphs and social networks*, pages 421–453. Springer.

Xiao, X. and Tao, Y. (2006). Personalized privacy preservation. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 229–240. ACM.

Xie, M., Yin, H., and Wang, H. (2006). An effective defense against email spam laundering. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, pages 179–190, New York, NY, USA. ACM.

Xie, Y., Yu, F., Achan, K., Panigrahy, R., Hulten, G., and Osipkov, I. (2008). Spamming botnets: signatures and characteristics. *SIGCOMM Comput. Commun. Rev.*, 38(4):171–182.

Xu, L., Chainan, S., Takizawa, H., and Kobayashi, H. (2010a). Resisting sybil attack by social network and network clustering. In *Proceedings of the 2010 10th IEEE/IPSJ International Symposium on Applications and the Internet*, SAINT '10, pages 15–21, Washington, DC, USA. IEEE Computer Society.

Xu, W., Zhang, F., and Zhu, S. (2010b). Toward worm detection in online social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, pages 11–20, New York, NY, USA. ACM.

Yang, Z., Wilson, C., Wang, X., Gao, T., Zhao, B. Y., and Dai, Y. (2011). Uncovering social network sybils in the wild. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, IMC '11, pages 259–268, New York, NY, USA. ACM.

Yu, H. (2011). Sybil defenses via social networks: A tutorial and survey. *SIGACT News*, 42(3):80–101.

Yu, H., Gibbons, P. B., Kaminsky, M., and Xiao, F. (2010). Sybillimit: a near-optimal social network defense against sybil attacks. *IEEE/ACM Trans. Netw.*, 18(3):885–898.

Yu, H., Kaminsky, M., Gibbons, P. B., and Flaxman, A. (2006). Sybilguard: defending against sybil attacks via social networks. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '06, pages 267–278, New York, NY, USA. ACM.

Zhang, C., Sun, J., Zhu, X., and Fang, Y. (2010). Privacy and security for online social networks: challenges and opportunities. *Network, IEEE*, 24(4):13–18.

Zheleva, E. and Getoor, L. (2008). Preserving the privacy of sensitive relationships in graph data. In *Privacy, security, and trust in KDD*, pages 153–171. Springer.

Zheleva, E. and Getoor, L. (2009). 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*, WWW '09, pages 531–540, New York, NY, USA. ACM.

Zheleva, E. and Getoor, L. (2011). *Privacy in social networks: A survey*, pages 277–306. Springer.

Zhou, B. and Pei, J. (2011). The k-anonymity and l-diversity approaches for privacy preservation in social networks against neighborhood attacks. *Knowledge and Information Systems*, 28(1):47–77.

Zhou, B., Pei, J., and Luk, W. (2008). A brief survey on anonymization techniques for privacy preserving publishing of social network data. *SIGKDD Explor. Newsl.*, 10(2):12–22.

Zhu, Z. and Cao, G. (2011). Applaus: A privacy-preserving location proof updating system for location-based services. In *INFOCOM, 2011 Proceedings IEEE*, pages 1889–1897. IEEE.

Zinman, A. and Donath, J. (2007). Is britney spears spam. In *Fourth Conference on Email and Anti-Spam, Mountain View, CA*.