Online Social Networks and Media 3–4 (2017) 1–21

Contents lists available at ScienceDirect

Online Social Networks and Media

journal homepage: www.elsevier.com/locate/osnem

Privacy and security in online social networks: A survey

Imrul Kayes[a,*], Adriana Iamnitchi[b]

[a] Sonobi Inc., 444 W New England Ave #215, Winter Park, FL 32789, USA
[b] Computer Science and Engineering, University of South Florida, Tampa, FL, USA

ARTICLE INFO

Article history:
Received 15 May 2017
Revised 10 August 2017
Accepted 18 September 2017

Keywords:
Privacy
Security
Online social networks

ABSTRACT

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.

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 2017, Facebook has 1.94 billion monthly active users and it is the third most visited site on the Internet [1]. Twitter, a social micro-blogging platform, claims over 313 million monthly active users, who send Tweets in more than 40 languages [2].

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 [3]; law enforcement organizations are gleaning evidence from OSNs to solve crimes [4]; activities on online social platforms change political regimes [5] and swing election results [6].

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 [7]. 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 [8].

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 [9] or employee fired for commenting on her salary compared with that of her boss [10], both on Facebook. On top of this, social networks themselves intentionally (e.g., Facebook Beacon controversy [11]) or unintentionally (e.g., published anonymized social data used for de-anonymization and inference attacks [12]) 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) 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 [13] or deploying automated social robots [14].

This article 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,

* Corresponding author.
E-mail address: imrulkayes11@gmail.com (I. Kayes).

https://doi.org/10.1016/j.osnem.2017.09.001
2468-6964/

such as privacy preserving social data publishing techniques [15], social graph-based techniques for mitigating Sybil attacks [16], OSN design issues for security and privacy requirements [17], or threats in OSNs [18], 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 countermeasures are available, and what are the challenges still to overcome.

2. A taxonomy of privacy and security problems in online social networks

A social network is an ecosystem consisting of a number of entities. These entities include, but not limited to, users, the OSN service provider, third-party applications, and advertisers. However, the primary stakeholders of this ecosystem are users (who receive various social networking services) and OSN providers (who provide those social networking services). The privacy and security problems bring significant consequences for users and OSN service providers. For users, potential consequences mean inappropriate sharing of personal information, i.e., leakage, and exploitation of personal details using active mining, e.g., information linkage [19]. For OSN services, privacy and security threats disrupt the proper functioning of the service and damage providers' reputation.

We propose a taxonomy of privacy and security problems in online social networks based on the stakeholders of the ecosystem and the axes from which privacy and security risks come. As we have already mentioned, we identify two primary stakeholders in online social networks: the OSN users and the OSN itself.

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 [20] show 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 [21]. 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 [22]. 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 [20]. Unintended revealing of personal information brings other online risks, including scraping and harvesting [23,24], social phishing [25], and automated social engineering [26].

In the ecosystem of an OSN, users interact with other users (a lot of them are complete strangers), use third-party social applications, and clicks on ads placed by the advertisers. Users' information leakage might happen to all of these entities. Moreover, users' data collected from multiple social networks lead to linkage problems, where a significantly broader profile of the user could be built by linking the user over the social networks.

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. However, attacks such as Sybil, DDoS, spam and malware on OSNs 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 categories (summarized in Table 1).

1. Leakages and linkages of user information and content: these issues relate to information disclosure threats. We identify a number of entities who are involved with users' information and their content leakage and linkage.
   (a) Leakages into 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 [27]), or may be crowdsourcing workers strolling and interacting with users for mischievous purposes [28]. Therefore, the challenge is to protect users and their information from other users.
   (b) Leakages into 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 [29].
   (c) Leakages into 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 [30] and most users do not have a real choice if they do not agree with the exchange. Consequently, the exploitation by the OSN of user's personal information is seen as a breach of trust, and many solutions have been proposed to hide personal information from the very service that stores it.
   (d) Linkages by aggregators: 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 [31]. Crawling users' data from multiple sites and multiple domains and further linking them 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 [32].
2. Attacks on the OSN: these attacks are aimed at the service provider itself, by threatening its core business. OSNs have been targeted by Distributed Denial-of-service (DDoS) attacks; have been used as platforms for propagating malware and social spam. These attacks can be performed by a number of ways. For example, attackers can create a number of Sybil identities and use them for spam content campaign or malware propagation. Attackers can also illegitimately take control of the accounts created by other users, and use those compromised accounts to launch an organized and planned attacks. Note that users of the platforms are

Table 1
Categories of privacy and security problems in OSNs.

Users' information and content leakages and linkages	Attacks on the OSN
Leakages into other users	Sybil attacks
Leakages into social applications	Compromised accounts
Leakages into the OSN	Social spam and malware
Linkages by aggregators	Distributed Denial-of-service attacks (DDoS)

also impacted by the attacks on the OSNs. However, attacks on the OSN exploit the social graph of the OSN and victimize more users by propagating rapidly. So, the priority of the service provider would be to identify and stop the propagation. In the following, we briefly discuss the actors (e.g., Sybils, compromised accounts) who could be used for attacks, and the actions (social spam, malware, DDoS attacks) of the actors.

(a) Sybil Attacks: Sybil attacks are characterized by users assuming multiple identities to manipulate the outcome of a service [33]. Not specific to OSNs, Sybil attacks were used, for example, to determine the outcome of electronic voting [34], to artificially boost the popularity of some media [35], or to manipulate social search results [36]. 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 [37].

(b) Attacks from compromised accounts: Compromised accounts are legitimate user accounts that are created and used by their fair owners, but have been compromised by attackers [38]. Unlike Sybil accounts, these accounts already have established social connections and normal social network usage history. But suddenly they are hacked by attackers and are later used for the ill purposes of the attackers.

(c) Social spam and malware: Social spam are contents or profiles that an OSN's "legitimate" users do not wish to receive [39]. 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. 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; frequently using social spam, e.g., [40].

(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 [41]. As many popular services, OSNs are also subjected to such coordinated, distributed attacks.

The rest of the paper is organized as follows. Mitigating leakages and linkages of user information and content (Sections 3–6) includes discussions of leakages into other users (Section 3), into social applications (Section 4), into the OSN itself (Section 5), and linkages by aggregators (Section 6). A summary of the mitigating solutions for the leakages and linkages of user information and content is shown in Fig. 1. Mitigating attacks on the OSN (Sections 7–11) includes a discussion of Sybil attacks (Section 7), attacks from compromised accounts (Section 8), social spam and malware (Section 9), and Distributed Denial-of-service attacks (Section 10). A summary of the mitigating solutions for the attacks on OSNs is presented in Fig. 2. Section 12 highlights some challenges and discusses future research directions. Finally, we conclude the paper in Section 13.

3. Mitigating attacks from other users

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 [42,43]) and 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 [44,45] argue that fine-grained privacy controls are crucial features for privacy management. Krishnamurthy and Wills [44] 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 [46] 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

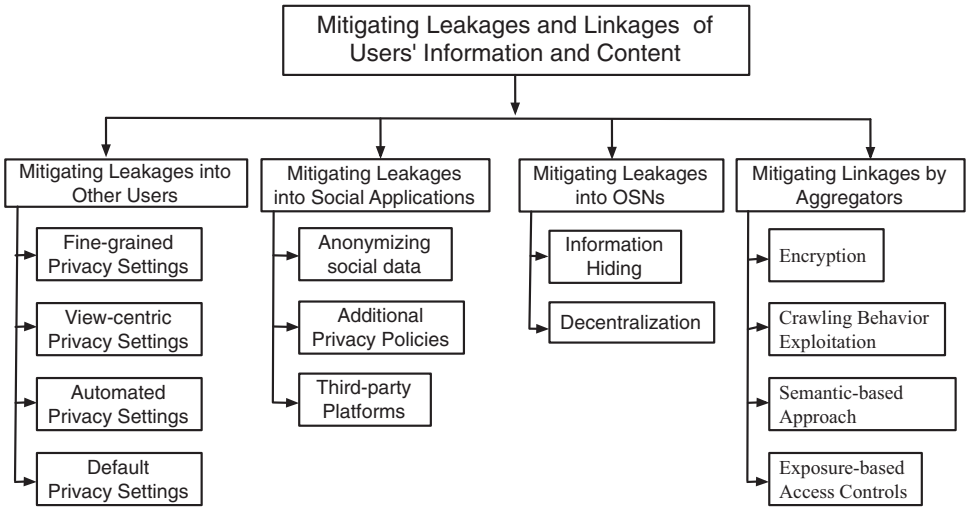

Fig. 1. Mitigating solutions for the users' leakage and linkage of formation and content.

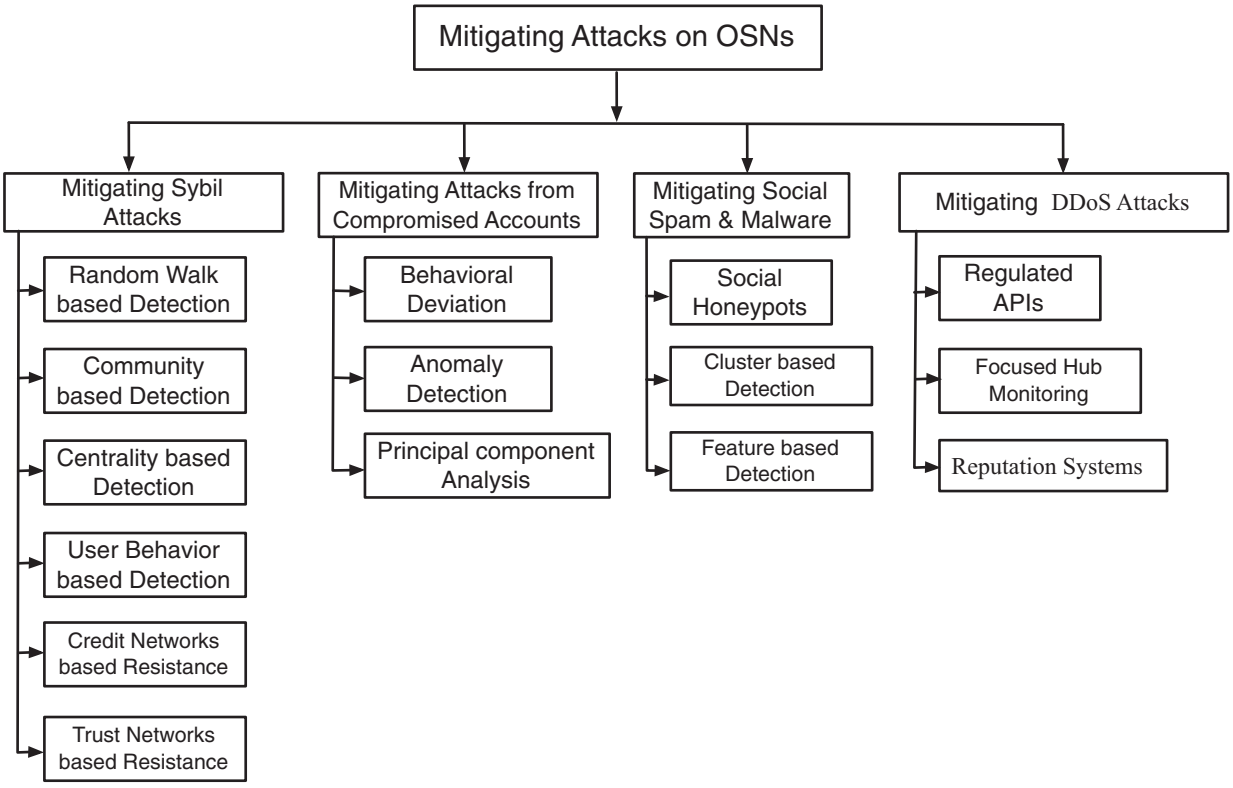

Fig. 2. Mitigating solutions for the attacks on OSNs.

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. [47] 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. [48] based on relationship type, degree of separation, and a quantification of trust between users in the network. Using regular expression notation Cheng et al. [49] propose user-to-user relationship-based access control (UURAC) model for OSNs. They consider relationship types, the number of hops on the path between users and resources, and the type of action to define the policies.

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 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 [50–52]. Although conceptually similar, Carminati et al. [51] provide 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 [52]. Rule-based privacy models have two challenges to overcome. 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 [53].

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 [54] 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. [55] 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 and Preibusch [56] 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 [57]. 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 [58], 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, its management is tedious for users with many groups. Wisniewskia et al. [59] propose privacy interface design based on privacy management strategies of the users. They use self-reported privacy behaviors of 308 Facebook users and classify six distinct privacy management strategies. These strategies are self explanatory from their names: privacy maximizers, selective sharers, privacy balancers, self-censors, time savers/consumers, and privacy minimalists. They advocate that these strategies could be used to personalize privacy recommendations.

A simpler interface is proposed by *C4PS* (Colors for Privacy Settings) [60], 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 recent work of Stern and Kumar [61] also have used colors for privacy settings and proposed a new interface in the shape of a wheel.

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. A qualitative study [62] of teenage OSN users shows that colorful privacy settings enable to have more control over the sharing of their information.

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* [63] 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 [64], an active learning paradigm, acknowledging the fact that users may quit labeling friends at any time. Bilogrevic et al. [65] also have employed machine learning techniques and proposed *SPISM* for privacy-aware information sharing in mobile social networks. Their system uses personal and contextual features and automatically defines what information to be shared with others and with what granularity.

Social circles [66] 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 [67]. While convenient, this approach limits users' flexibility in changing the automate settings.

Danezis [68] 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.

Yuan et al. [69] also propose context-dependent automated privacy settings for photo sharing OSNs. Their proposed model uses the semantics of the photo and requestor's contextual information to define whether an access to the photo will be granted or not at a certain context.

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 savvy user will want more transparency and possibly more control. To this end, the privacy wizard [63] 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 [70]. Similarly, the majority of Facebook users has default settings [44,71,72]. Under-utilization of privacy options are mostly due to poor privacy setting interface [58], intricate privacy settings [73], and inherent trust in OSNs [72,74]. The problem with not changing the default settings is that they almost always tend to be more open that the users would prefer [75]. To overcome this situation, approaches to automatically generate more appropriate default privacy settings have been proposed.

PriMa [76] 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 parameterized tuning contribute to longer policy generation and enforcement time. A related approach, *PolicyMgr* [77], uses supervised learning of user-provided example policy settings and builds classifiers that are then used for automatically generating access control policies.

Aegis [78,79] is a privacy framework and implementation that leverages the *'Privacy as Contextual Integrity'* theory proposed by Nissenbaum [32] 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* (a social game) and *Horoscopes* (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 Fig. 3. 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 markup 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 Fig. 3), 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 [80].

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 [81,82] 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 [81], violating the developer policies that private information of friends are not accessible. The Wall Street Journal finds evidence that Facebook applications transmit identifying information to advertising and tracking companies [82].

Third, 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 [29] 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 [83], which states that every user should only get the minimal set of access rights that enables him to complete his task.

Finally, a poorly designed API might lead to application impersonation attacks, where an attacker successfully assumes the identity of a legitimate application and possess users' data shared with the application. For example, recently, many OSNs use *OAuth 2.0* protocol to grant access to API endpoints. Hu et al. [84] show that the application impersonation attack is possible due to OAuth's multiple authorization flows and token types. Their investigation on 12 major OSN providers show that 8 of them are vulnerable to application impersonation attacks.

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 [29] 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

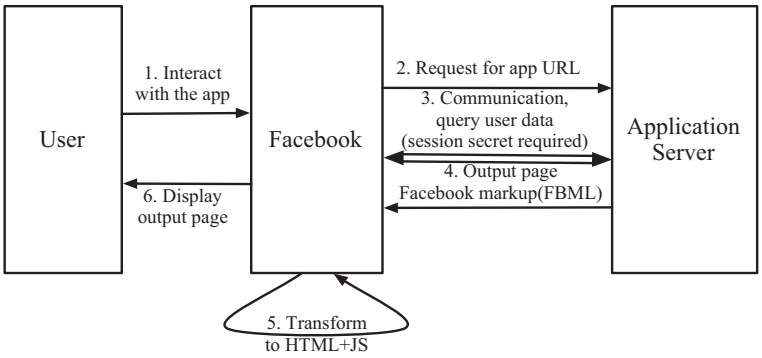

Fig. 3. Data flow in a Facebook application.

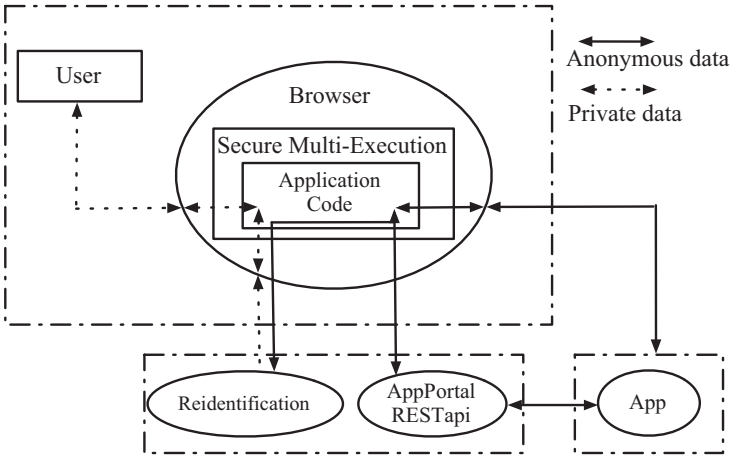

Fig. 4. Data flow in a PESAP aware browser [85].

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 [85] 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 symmetric 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 [86], 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. Fig. 4 shows the data flow in a PESAP-aware browser.

4.2. Enforcing additional privacy policies to social applications

Besmer et at. [87] 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.

Similar to the previous work, Cheng et al. [88] also propose an access control framework. However, Besmer's et at. approach allows applications to transmit users' data to their servers. On the contrary, Cheng's et al. framework only permits privacy-nonsensitive data to be transmitted, if any functionality of the application runs outside of the OSN. Applications (or functions of an

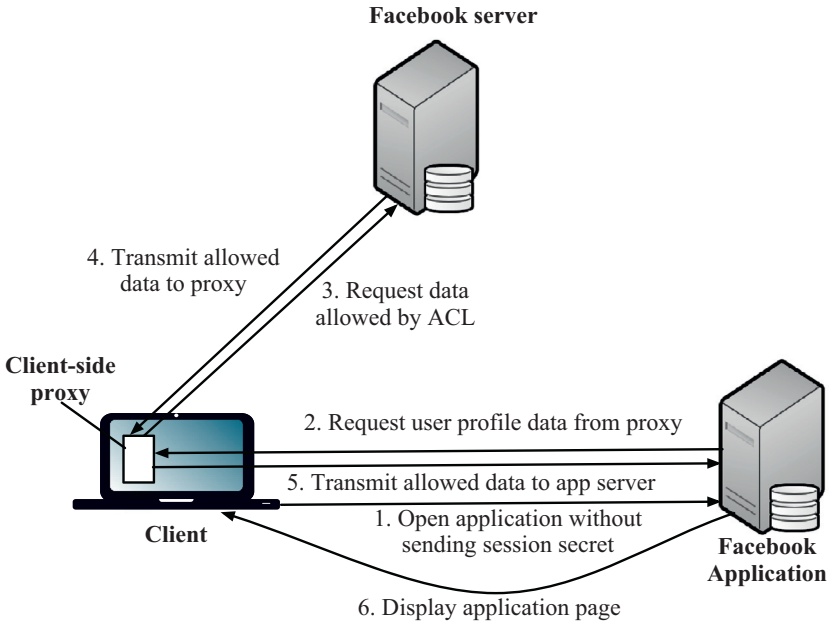

Fig. 5. A data-flow between applications and server with PoX [90].

application) that run under the surveillance of the OSN can only get the raw private data. Kavianpour et al. [89] propose to classify social applications based on authorization rights of the applications. They analyze the data transfer between a set of applications and Facebook and classify the applications based on extracted features of data transfer. They show that their authorization-based framework is able to reduce the risk of data leakage to third-party applications.

4.3. Third-party platforms for running social applications

Egele et al. [90] 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 Fig. 5. 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.

xBook [91] 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. A recent and similar approach of xBook is MUTT [92], however, it has the challenges related to xBook to overcome.

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 May 2017, we found 3048 words on Instagram's privacy policies and 3806 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 [93], 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 [30].

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 [72] 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 [94] 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 message, 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 [95] 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 [96] 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 [97], 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 ciphertext, decrypts it, and substitutes the real data for the fake data. If the value is not found, then the data is left unchanged. 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.

Virtual Private Social Networks) (VPSN) [98], 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 [99])

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 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. In fact, a recent user study [100] finds that higher usability costs, lack of trust, and poor performance are the main causes of poor or no adoption of these services.

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 [101] 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 [7,102] 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 [103] is another P2P-based OSN. It implements common functionalities in OSNs using OSGi-based 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 [104] is a framework for the implementation of a P2P based OSN on a Likir DHT [105]. 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-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 [106] 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 [107,108] 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 still an alternative for centralized OSNs, specially for privacy-concerned users. For example, Diaspora (https://joindiaspora.com/) 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 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 [31]. For example, crawling 100 million public profiles from Facebook created news recently [109]. 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" [110].

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. 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 [111]. 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 [112]. 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 [112] and crawler's observational behavior [113] 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 [112]. 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 [112] 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.

PUBCRAWL [114] is based on the observation that the traffic that a crawler generates is significantly different from fair users. It uses content-based and timing-based footprints to distinguish crawler traffic from regular traffic. Content-based features are extracted from URLs (e.g., access errors, page revisits) and HTTP headers (e.g., cookies, referrers). Timing-based features are obtained from the analysis of the time series produced by the stream of requests. Finally, PUBCRAWL relies on machine learning techniques and trains classifiers using the features that can separate crawler traffic from user traffic.

Wan [115] uses URLs to develop an anti-crawler system called PathMarker. PathMarker differentiates fair users from malicious crawlers by using the URL visiting path and URL visiting timing features that can be obtained from the URL.

Genie [113] exploits browsing patterns of honest/real users and crawlers and thwarts large-scale crawls using Credit Networks [116,117]. While PUBCRAWL uses physical network layer differences, Genie uses social network layer differences. Genie's 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 viewer; 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.

7. Mitigating Sybil attacks

The Sybil attack is a fundamental problem in distributed systems. The term *Sybil* was first introduced by Douceur [33], 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,

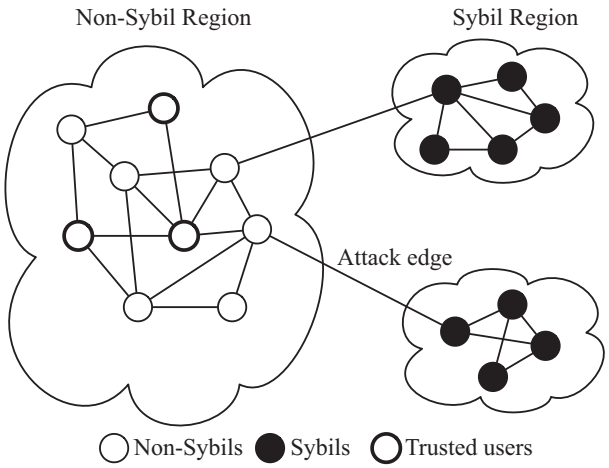

Fig. 6. The system model for Sybil detection.

Facebook and BitTorrent have become vulnerable to Sybil attacks. For example, Facebook anticipates that up to 83 million of its users may be illegitimate [118], which is far more than what it anticipates (54 million) earlier [119]. Researchers found that Sybil users affect the correct functioning of the system by contributing malicious contents [36,120] and illegitimately increasing influence and power [35,121].

Malicious activities from Sybil users are posing serious threats to OSN users, who trust the service and depend on it for online interactions. 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 [122].

Two categories of solutions are available to defend Sybils: Sybil detection and Sybil resistance. Sybil detection schemes [37,122–125] 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 [126,126,127] (Section 7.2). In a tutorial and survey Yu [16] 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 Fig. 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 Fig. 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 [128,129]. 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 [37,122,125,130], social community [131], and network centrality [132] to detect Sybils in the network. SybilGuard [37] is a decentralized Sybil detection scheme, which uses Assumptions 1–3. 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 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 is 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 [130] assumptions are also Assumptions 1–3. 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 [122] 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 [37,133], it uses power iteration [134] in calculating the landing probability of random walks.

Boshmaf et al. [135] use both social graph and users' behavioral differences to identify Sybils. They first use user-level activities such as number of friends, interaction frequency and volume of each user to potential victims i.e., honest users that are likely to accept friend requests from Sybils. Then they annotate the social graph by introducing lower weights to edges incident to potential victims. Finally, starting from a seed node they propagate trust using power iterations and rank users. As Sybil users are usually connected to honest users, they earn lower trust values.

However, one potential problem with all of the previous approaches is that they do not tolerate noise in the prior knowledge about known non-Sybil or Sybil nodes. SybilBelief [125], another random walk-based semi-supervised learning framework, overcomes the limitation. SybilBelief propagates Sybil/non-Sybil label information from known Sybil/non-Sybil labels to the remaining nodes in the system by modeling the social network as Markov Random Fields. However, the twist is, a portion of these known Sybil/non-Sybil labels might be inaccurate, which earlier approaches fail to address.

Viswanath et al. [131] 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. [132] 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 [136]. The scheme assumes that the number of attack edges is limited and Sybil and 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 (e.g., [137]). For example, the assumption that Sybils' have problems in creating social connections with legitimate users (non-Sybils) is not well established. Although study [138] 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 [26,139,140] 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. Social graph-based Sybil detection techniques are vulnerable to such adversarial social engineering attacks. So, recently researchers have focused on combining user-level activity footprint and graph-level structures (e.g., Íntegro [141], VoteTrust [142]).

Also, another assumption that a social network is fast-mixing may not be right for all social networks. Study [143] 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.

Despite considerable research effort on social graph based approaches, they are far achieving desired goals. By designing simple attack strategies Koll et al. [144] recently show that an attacker could launch attacks to circumvent social graph based solutions.

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 [145,146], spam protection [147], online auctions [126], reputation systems [148], and collaborative mobile applications [149].

Note two assumptions of Sybil detection schemes: (1) non-Sybil region is fast mixing, (2) Sybils cannot 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 [126,127,147] share a common approach in resisting Sybils—they use a *credit network* built on the top of the social network of users [150]. Originally proposed in the electronic commerce community, *Credit Networks* [116,117] 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 Fig. 7, which is a core defense philosophy of some resistance schemes. In the following, we provide a brief overview of the Sybil resistance schemes.

Ostra [147] 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

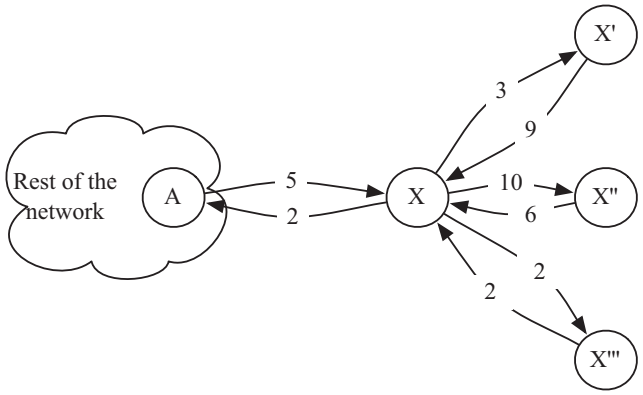

Fig. 7. Credit network based Sybil resistance [150]. The network contains four Sybil identities as nodes X, X', X'', X''' of a Sybil user. A directed edge (X, Y) represents how much credit is available to X from Y. If X wants to pay credits from other three nodes, the credits must be deducted from X's single legitimate link to A. So, a Sybil's other identities do not provide any additional credits in the rest of the network.

send a large number of unwanted communications, unless she also has additional trust relationships.

Bazaar [126] 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 [127] 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 [151,152]. 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.

TrueTop [153] is a sybil-resilient system for Twitter which measures the influence of Twitter users given the presence of Sybils. It is based on two observations: (1) non-Sybil users might follow strangers, but they are more careful and selective in retweeting, replying to, and mentioning other users; (2) influential users get much more retweets, replies, and mentions compared to other users. TrueTop first constructs an interaction graph using users and their retweets, replies, and mentions. Then it does iterative credit distribution in the interaction graph and finally selects top-K influential users.

MobID [149] 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. Attacks from compromised accounts

A compromised account is a legitimate account that has been hacked and taken over by an attacker [38]. The attacker can exploit the account for a number of mischievous purposes, such as, spreading contents via wall posts or direct messages, liking commercial social pages, and following others. Note that compromised accounts are different than Sybil or fake accounts in that compromised accounts have an established trust relationship with others. So, they are not deemed suspicious to OSNs. Attackers exploit this embedded trust and use the accounts for increasing influence and power.

User accounts can be compromised in a number of ways. Users might trust third-party websites or applications with their OSN credentials and those third-parties might be malicious. Users' account passwords are sometimes weak and bots could guess them. Attackers also use cross-site scripting and social phishing to compromise users' accounts.

Compromised accounts have negative consequences on the OSN. They damage the reputation of the system by providing fake like, following and promoting unwanted content. Victims of the compromised accounts lose their accounts and hence their social connections. A research [154] on Twitter compromised accounts shows that about 27% of the compromised users change to a new account once their accounts are compromised.

A number of solutions [38,155,156] have been proposed to detect compromised accounts in OSNs. These solutions exploit behavioral deviation of the account before and after an account is compromised. Compa [38] detects compromised accounts using statistical modeling of user behavior and anomaly detection. It makes two assumptions—(1) a compromised account will show noticeable behavioral differences compared to the behavior of the legitimate owner of the account, and (2) an attacker will spread the same malicious content (e.g., tweet or messages) from a subset of account it has compromised. Compa makes behavioral profile of a user and checks for a sudden and significant violation of disseminated content from the profile. It makes the behavioral profile of a user considering content features, such as time of posting, the source of the content (e.g., third-party applications), language, links of the content, interaction and proximity to other users. Then it computes an anomaly score for a new content comparing it to the user's already established profile. The user is put in a suspicious category if a significant portion of new messages has higher anomaly scores. Finally, Compa groups similar content and hence compromised accounts using two similarity measures: content similarity and url similarity. One potential problem with Compa is that attackers can still dodge the system by not posting the same messages from the accounts it has compromised.

SynchroTrap [155] also assumes that the compromised accounts act together in different social network contexts (e.g., page like, photo upload, follow others). It further assumes that those actions are loosely synchronized. SynchroTrap is a more generalized

version of Compa for any social network context, as it assumes that actions (e.g., photo upload) are only coordinated, the action might be content independent (e.g., follow others). However, SynchroTrap makes the real difference in terms of scalability. As it is built as an incremental processing system, it can efficiently process massive OSN user activity data. (The system was successfully deployed on Facebook and Instagram). First, SynchroTrap abstracts users' actions using tuples—a combination of user ID, timestamp of actions, type of action (e.g., posting), IP address of the users. Then, for each pair of users, using Jaccard similarity, it computes similarity between two users for their actions during a period of time. Finally, it uses hierarchical clustering algorithms to group users having the similar actions.

Viswanath et al. [156] also uses anomaly detection techniques to detect compromised accounts (anomalous behavior in general). But the difference with Compa is that it does not make any assumptions about attack strategy (e.g., Compa assumes coordinated posting of the same content from compromised accounts). They focus on modeling Facebook *Like* activity behavior of normal users. In so doing, they use features, such as, *temporal* (number of likes per day), *spatial* (number of likes in different categories), and *spatio-temporal* (summary of the distribution of like categories using entropy). Of these features, they use principal component (PCA) analysis technique to detect the features that best explain normal user behavior. They experimented with Facebook, Yelp and Twitter datasets and found that three to five principal components are enough to model normal users' behavior. These components are later used to flag anomalous users, whose behavior do not fit the components. Using ground truth data from Facebook compromised users, the authors showed that the technique worked well.

Ruan et al. [157] introduce a set of social behavioral features that can be used to measure behavioral deviation when an account is compromised. More specifically they use eight new behavioral features which cover both a user's extroversive posting and introversive browsing activities. Using sample data from Facebook they show that social behavioral profile can effectively differentiate users with accuracy up to 98.6%.

9. Mitigating social spam

Spam is a news in web-based systems (e.g., [158,159]). However, OSNs have added a new flavor to it by acting as effective tools for spamming activities and propagation. Social spam (e.g., [160,161]) is unwanted content that is directed specifically at users of the OSN. The worst consequences of social spam include phishing attacks [25] and malware propagation [162].

Spamming activity is pervasive in OSNs and spammers are successful. For example, about 0.13% of spam tweets in Twitter generate a page visit [120], which is only 0.003–0.006% for spam email [163]. 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 [26] 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 [164] (Section 9.2).

9.1. Spam content and profile detection

Some early spam profile detections [165–167] used social honeypots. A honeypot is a trap deployed to capture examples of nefarious activities in networked systems [165]. For years, researchers have used honeypots to characterize malicious hacker activities [168], to obtain footprints of email address crawlers [169], and to create intrusion detection signatures [170]. Social honeypots are used to monitor spammers' behaviors and store their information from the OSNs [167].

Webb et al. [166] took the first step to characterize spam in OSNs using social honeypots. They created honeypot MySpace profiles in different geographic locations for harvesting deceptive spam profiles on MySpace. An automated program (commonly known as bots) worked on behalf of a honeypot profile and collected all of the traffic it received (via friend requests). After four months of the deployment and operation, the bots collected a representative sample of friend requests (and corresponding spam profiles). Through statistical analysis the authors showed 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 [167], 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. These features are later used to train machine learning classifiers that are able to distinguish spam and fair profiles.

One of the limitations of these honeypot-based solutions [166,167] 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. As such, recent research on honeypot-based spam detection is focusing more on how to build more effective social honeypots (e.g., [171]). Another problem is that, 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's et al. solution [172] overcomes some limitations of the previous two honeypot-based papers by using richer feature sets. 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). 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. Finally, the authors manually inspected and labeled profiles as spam and used *Random Forest* algorithm for classification.

Benevenuto et al. [173] 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.

Kayes et al. [174] identify abusive content providers in community question answering social networks. Similar to the previous approaches, they have used a number of platform-related features to train machine learning classifiers. But the difference is that they not only used users' social network and activity-specific features, but also leveraged the crowd-sourced rule violations reports contributed by the members of the network.

In a recent article Cresci et al. [175] show a paradigm-shift of social spam profiles. They find evidence of a new generation of spambots, so-called social spambots. These bots are intelligent and evolve over time. For example, they can search the Internet to fill their profiles with real data, post credible sources of information, and interact socially with friends or followers. Cresci et al. show that traditional spam profile detection techniques such as those based on textual features of shared messages, posting patterns and social relationships do not work for these social spambots as the bots demonstrate human-like behavior. Researchers have used statistical divergence [176], behavioral DNA sequencing [177], and graph anomaly [178–180] to detect spambots. Ferrara et al. [181] have done a comprehensive study on detecting the spambots. In this paper, we also describe few of the techniques.

Viswanath et al. [176] detect tamper in crowd computation using statistical difference of users participating in the computation. The intuition behind their detection is that statistical distribution of reputation scores such as number of followers or friends of the users participating in a tampered crowd computation significantly differs from an untampered computation. They use Kullback–Leibler distance between the distributions and mark a crowd computation tampered if the distance exceeds a threshold.

Cresci et al. [177] use DNA-inspired behavioral modeling to detect spambot groups. They associate each account to a digital DNA sequence by encoding the behavior of the account in a string. These accounts are then compared against one another in order to obtain anomalous similarities using Longest Common Substring (LCS). Finally they label the accounts as spambots if the accounts share a suspiciously long DNA substring.

9.2. Spam campaigns detection

Chu [182] 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. [164] 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 [183] 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. Using threshold filters on these two properties they found clusters of wall posts and classified them as potentially malicious spam campaigns. Template-based spam campaign detection has been also done in Twitter [184].

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 [41]. A Distributed Denial-of-service attack (DDoS) deploys multiple attacking entities to simultaneously launch the attack (we refer readers [185] 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 [186]. 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. [187] 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 [188] 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 [40]. 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. [189] 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., [190]) cannot be applied to them. In the context of OSNs, Xu et al. [191] 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. Challenges and future research directions

As the OSNs are enjoying unprecedented popularity, keeping users engaged with new functionalities, new privacy and security threats are emerging [192,193]. The dynamic landscape of privacy and security attacks have enabled researchers to continuously looking forward for new threats and provide mitigating techniques. However, there are open problems that still withstand the plethora of solutions in the literature. An overview of the problems is presented below.

The privacy solutions reviewed in Section 3 are focused on specific aspect of privacy, such as, enabling granular settings, providing visual feedback through designing user friendly and informed graphical interface, or generating automated or default privacy policies. However, an integrated privacy solution covering all the planes is still expected. Such a solution might face multiple challenges, e.g., too much granular privacy settings would be a problem in designing succinct interfaces, automated or default privacy policies might have different interface requirements. Moreover, automated and default privacy solutions also have bootstrapping problems to overcome. A newly joined OSN user has no history of interactions that could be used as an input of automation.

A body of literature has used third-party platforms for protecting users from social applications (Section 4) and from OSNs (Section 5.1). The third-party platforms have been used for appropriate norm following execution of social applications and limiting the transfer of the social data from applications to other parties. Third-party platforms have been also used to hide users' real data to protect them from the OSN. However, those solutions themselves have to be trusted by users and by applications, as they are expected to protect users' personal data and enable a third-party application's execution. Moreover, research needs to propose promising business models for those platforms, because hosting and executing applications on those platforms have a high requirement of logistics and maintenance.

One possible future research direction includes understanding the privacy leakage and associated risks when OSNs work as a Web tracker. OSNs (e.g., Facebook, Twitter) continue to be the login of choice for many websites and applications. As such, OSNs can track their users in third-party websites by placing cookies to users' devices on behalf of those websites. Note that OSNs already know what users do in their platforms. Tracking the users in third-party websites enables them to create a more detailed user profile. As such, OSNs could essentially work as a traditional third-party Web aggregator by offering advertisers targeted advertising in publishers' websites. In general, third-party Web tracking has seen much policy debate recently [194–196], and OSNs have aggravated the tracking. Research could explore a comprehensive risk assessment and solutions considering OSNs as potential trackers.

The attacks discussed in this article 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 prerequisite 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. Researchers still need to explore the attacks that are are synergies of attacks.

13. 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.

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 [197,198]. Moreover, real-world rewards and discounts give incentives for users in LSP to cheat on their locations, and hence research [199,200] 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 [8].

References

[1] Zephoria, The Top 20 Valuable Facebook Statistics, Zephoria2017, URL: https://zephoria.com/top-15-valuable-facebook-statistics/.

[2] Twitter, Twitter Usage/Company Facts, Twitter2017, URL: https://about.twitter.com/company.

[3] E. Protalinski, 56% of employers check applicants' Facebook, LinkedIn, Twitter, 2012, URL: http://www.zdnet.com/article/56-of-employers-check-applicants-facebook-linkedin-twitter/.

[4] H. Kelly, Police embrace social media as crime-fighting tool, 2012, URL: http://www.cnn.com/2012/08/30/tech/social-media/fighting-crime-social-media.

[5] G. Lotan, E. Graeff, M. Ananny, D. Gaffney, I. Pearce, et al., The arab spring — the revolutions were tweeted: Information flows during the 2011 tunisian and egyptian revolutions, Int. J. Commun. 5 (2011) 31.

[6] P. Jha, Facebook users could swing the results in 160 Lok Sabha constituencies, 2013, URL: http://www.thehindu.com/news/national/facebook-users-could-swing-the-results-in-160-lok-sabha-constituencies/article4607060.ece.

[7] L. Cutillo, R. Molva, T. Strufe, Safebook: a privacy-preserving online social network leveraging on real-life trust, IEEE Commun. Mag. 47 (12) (2009) 94–101.

[8] H. Nissenbaum, A contextual approach to privacy online, Daedalus 140 (4) (2011) 32–48.

[9] W.B. Dam, School teacher suspended for Facebook gun photo, 2009, URL: http://www.foxnews.com/story/2009/02/05/schoolteacher-suspended-for-facebook-gun-photo/.

[10] D. Mail, Bank worker fired for Facebook post comparing her 7-an-hour wage to Lloyds boss's 4000-an-hour salary, 2011, URL: http://dailym.ai/fjRTlC.

[11] C. Dwyer, Privacy in the age of Google and Facebook, IEEE Technol. Soc. Mag. 30 (3) (2011) 58–63.

[12] A. Narayanan, E. Shi, B.I. Rubinstein, Link prediction by de-anonymization: how we won the Kaggle social network challenge, in: Proceedings of the 2011 International Joint Conference on Neural Networks (IJCNN), IEEE, 2011, pp. 1825–1834.

[13] E. Staff, Verisign: 1.5 m Facebook accounts for sale in web forum, 2010, URL: http://www.pcmag.com/article2/0,2817,2363004,00.asp.

[14] C. Wagner, S. Mitter, C. Körner, M. Strohmaier, When social bots attack: modeling susceptibility of users in online social networks, in: Proceedings of the 2012 International Conference on World wide web (WWW), vol. 12, 2012.

[15] E. Zheleva, L. Getoor, Privacy in Social Networks: A Survey, Springer, pp. 277–306.

[16] H. Yu, Sybil defenses via social networks: a tutorial and survey, SIGACT News 42 (3) (2011) 80–101, doi:10.1145/2034575.2034593.

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

[18] M. Fire, R. Goldschmidt, Y. Elovici, Online social networks: threats and solutions, IEEE Commun. Surv. Tutor. 16 (4) (2014) 2019–2036.

[19] B. Krishnamurthy, Privacy and online social networks: can colorless green ideas sleep furiously? IEEE Secur. Priv. 11 (3) (2013) 14–20.

[20] R. Gross, A. Acquisti, Information revelation and privacy in online social networks, in: Proceedings of the ACM Workshop on Privacy in the Electronic Society, ACM, 2005, pp. 71–80.

[21] T. Hansen, Social media gives stalkers unprecedented access to victims, 2015, URL: http://www.mcphersonsentinel.com/article/20150112/NEWS/150119927.

[22] L. Sweeney, Uniqueness of Simple Demographics in the US population, Carnegie Mellon University, Laboratory for International Data Privacy (2000).

[23] J. Lindamood, R. Heatherly, M. Kantarcioglu, B. Thuraisingham, Inferring private information using social network data, in: Proceedings of the Eighteenth International Conference on World Wide Web, ACM, 2009, pp. 1145–1146.

[24] T. Strufe, Profile popularity in a business-oriented online social network, in: Proceedings of the Third Workshop on Social Network Systems, ACM, 2010, pp. 2:1–2:6.

[25] T.N. Jagatic, N.A. Johnson, M. Jakobsson, F. Menczer, Social phishing, Commun. ACM 50 (10) (2007) 94–100.

[26] L. Bilge, T. Strufe, D. Balzarotti, E. Kirda, All your contacts are belong to us: automated identity theft attacks on social networks, in: Proceedings of the Eighteenth International Conference on World Wide Web, ACM, 2009, pp. 551–560.

[27] T. Hwang, I. Pearce, M. Nanis, Socialbots: voices from the fronts, Interactions 19 (2) (2012) 38–45, doi:10.1145/2090150.2090161.

[28] G. Stringhini, G. Wang, M. Egele, C. Kruegel, G. Vigna, H. Zheng, B.Y. Zhao, Follow the green: growth and dynamics in twitter follower markets, in: Proceedings of the 2013 Conference on Internet Measurement Conference, IMC '13, ACM, New York, NY, USA, 2013, pp. 163–176, doi:10.1145/2504730.2504731.

[29] A. Felt, D. Evans, Privacy protection for social networking APIs, in: Proceedings of the 2008 Web 2.0 Security and Privacy, 2008.

[30] C. Fiesler, A. Bruckman, Copyright terms in online creative communities, in: Proceedings of the Annual Conference Extended Abstracts on Human Factors in Computing Systems, CHI'14, ACM, 2014, pp. 2551–2556.

[31] J. Bonneau, J. Anderson, G. Danezis, Prying data out of a social network, in: Proceedings of the 2009 International Conference on Advances in Social Network Analysis and Mining, 2009, pp. 249–254.

[32] H. Nissenbaum, Privacy as contextual integrity, Wash. Law Rev. 79 (1) (2004) 119–158.

[33] J. Douceur, The Sybil attack, in: Proceedings of the Peer-to-Peer Systems, Springer Berlin, Heidelberg, 2002, pp. 251–260.

[34] D. Riley, Stat gaming services come to YouTube, 2007, URL: http://www.bbc.co.uk/news/technology-18813237.

[35] J. Ratkiewicz, M. Conover, M. Meiss, B. Gonçalves, A. Flammini, F. Menczer, Detecting and tracking political abuse in social media, in: Proceedings of the 2011 International Conference on Weblogs and Social Media, ICWSM, 2011.

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

[37] H. Yu, M. Kaminsky, P.B. Gibbons, A. Flaxman, Sybilguard: defending against Sybil attacks via social networks, in: Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, ACM, 2006, pp. 267–278.

[38] M. Egele, G. Stringhini, C. Kruegel, G. Vigna, Compa: detecting compromised social network accounts, in: Proceedings of the 2013 Symposium on Network and Distributed System Security (NDSS), 2013.

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

[40] Facebook, Facebook's Continued Fight Against Koobface, Facebook2012, URL: http://on.fb.me/y5ibe1.

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

[42] L. Banks, S. Wu, All friends are not created equal: an interaction intensity based approach to privacy in online social networks, in: Proceedings of the 2009 International Conference on Computational Science and Engineering, 2009, pp. 970–974.

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

[44] B. Krishnamurthy, C.E. Wills, Characterizing privacy in online social networks, in: Proceedings of the First Workshop on Online Social Networks, 2008, pp. 37–42.
[45] A. Simpson, On the need for user-defined fine-grained access control policies for social networking applications, in: Proceedings of the 2008 Workshop on Security in Opportunistic and SOCial Networks, ACM, 2008, pp. 1:1–1:8.
[46] S. Kruk, FOAM-Realm: control your friends access to the resource, in: Proceedings of the First Workshop on Friend of a Friend, 2004.
[47] H.C. Choi, S.R. Kruk, S. Grzonkowski, K. Stankiewicz, B. Davis, J. Breslin, Trust models for community aware identity management, in: Proceedings of the 2006 Identity, Reference and Web Workshop, in Conjunction with WWW, 2006, pp. 140–154.
[48] B. Carminati, E. Ferrari, A. Perego, Rule-based access control for social networks, in: Proceedings of the 2006 International Conference on On the Move to Meaningful Internet Systems, 2006, pp. 1734–1744.
[49] Y. Cheng, J. Park, R. Sandhu, An access control model for online social networks using user-to-user relationships, IEEE Trans. Dependable Secur. Comput. 13 (4) (2016) 424–436.
[50] N. Elahi, M. Chowdhury, J. Noll, Semantic access control in web based communities, in: Proceedings of the Third International Multi-Conference on Computing in the Global Information Technology, 2008, pp. 131–136.
[51] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, B. Thuraisingham, A semantic web based framework for social network access control, in: Proceedings of the Fourteenth ACM Symposium on Access Control Models and Technologies, SACMAT '09, 2009.
[52] A. Masoumzadeh, J. Joshi, Ontology-based access control for social network systems., IJIPSI 1 (1) (2011) 59–78.
[53] R. Engelmore (Ed.), Readings from the AI Magazine, American Association for Artificial Intelligence, Menlo Park, CA, USA, 1988.
[54] P.W. Fong, Relationship-based access control: protection model and policy language, in: Proceedings of the First ACM Conference on Data and Application Security and Privacy, 2011, pp. 191–202.
[55] F. Giunchiglia, R. Zhang, B. Crispo, RelBAC: relation based access control, in: Proceedings of the Fourth International Conference on Semantics, Knowledge and Grid, 2008, pp. 3–11.
[56] J. Bonneau, S. Preibusch, The privacy jungle: on the market for data protection in social networks, in: Proceedings of the 2010 Economics of Information Security and Privacy, Springer, 2010, pp. 121–167.
[57] K. Strater, H.R. Lipford, Strategies and struggles with privacy in an online social networking community, in: Proceedings of the Twenty-second British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction, British Computer Society, 2008, pp. 111–119.
[58] H.R. Lipford, A. Besmer, J. Watson, Understanding privacy settings in Facebook with an audience view., UPSEC 8 (2008) 1–8.
[59] P. Wisniewski, B. Knijnenburg, H.R. Lipford, Making privacy personal: profiling social network users to inform privacy education and nudging, Int. J. Hum.-Comput. Stud. 98 (2017) 95–108.
[60] T. Paul, M. Stopczynski, D. Puscher, M. Volkamer, T. Strufe, C4PS – helping facebookers manage their privacy settings, in: Proceedings of the 2012 Social Informatics, 2012, pp. 188–201.
[61] T. Stern, N. Kumar, Improving privacy settings control in online social networks with a wheel interface, J. Assoc. Inf. Sci. Technol. 65 (3) (2014) 524–538.
[62] M. van der Velden, M. Machniak, Colourful privacy: designing visible privacy settings with teenage hospital patients, in: Proceedings of the Sixth International Conference on Information, Process, and Knowledge Management, 2014.
[63] L. Fang, K. LeFevre, Privacy wizards for social networking sites, in: Proceedings of the Nineteenth International Conference on World Wide Web, ACM, 2010, pp. 351–360.
[64] D.D. Lewis, W.A. Gale, A sequential algorithm for training text classifiers, in: Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Springer-Verlag, New York, Inc., 1994, pp. 3–12.
[65] I. Bilogrevic, K. Huguenin, B. Agir, M. Jadliwala, M. Gazaki, J.-P. Hubaux, A machine-learning based approach to privacy-aware information-sharing in mobile social networks, Pervasive Mob. Comput. 21 (2015) 1–18.
[66] F. Adu-Oppong, C.K. Gardiner, A. Kapadia, P.P. Tsang, Social circles: tackling privacy in social networks, in: Proceedings of the 2008 Symposium on Usable Privacy and Security (SOUPS), 2008.
[67] N. Mishra, R. Schreiber, I. Stanton, R.E. Tarjan, Clustering social networks, in: Proceedings of the Fifth International Conference on Algorithms and Models for the Web-Graph, Springer-Verlag, 2007, pp. 56–67.
[68] G. Danezis, Inferring privacy policies for social networking services, in: Proceedings of the Second ACM Workshop on Security and Artificial Intelligence, ACM, 2009, pp. 5–10.
[69] L. Yuan, J. Theytaz, T. Ebrahimi, Context-dependent privacy-aware photo sharing based on machine learning, ICT Systems Security and Privacy Protection: Proceedings of the Thirty-second IFIP TC 11 International Conference, Springer, 2017, pp. 93–107.
[70] B. Krishnamurthy, P. Gill, M. Arlitt, A few chirps about twitter, in: Proceedings of the First Workshop on Online Social Networks, 2008, pp. 19–24.
[71] R. Gross, A. Acquisti, Information revelation and privacy in online social networks, in: Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, 2005, pp. 71–80.
[72] A. Acquisti, R. Gross, Imagined communities: awareness, information sharing, and privacy on the Facebook, in: Privacy Enhancing Technologies, Springer, 2006, pp. 36–58.
[73] M. Madejski, M.L. Johnson, S.M. Bellovin, The Failure of Online Social Network Privacy Settings, Department of Computer Science, Columbia University (2011).
[74] D. Boyd, Friendster and publicly articulated social networking, in: Proceedings of the 2004 Extended Abstracts of the Conference on Human Factors and Computing Systems (CHI 2004), 2004, pp. 1279–1282.
[75] Y. Liu, K.P. Gummadi, B. Krishnamurthy, A. Mislove, Analyzing Facebook privacy settings: user expectations vs. reality, in: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, 2011, pp. 61–70.
[76] A.C. Squicciarini, F. Paci, S. Sundareswaran, Prima: a comprehensive approach to privacy protection in social network sites, Ann. Telecommun.-Ann. Télécommun. 69 (1–2) (2014) 21–36.
[77] M. Shehab, G. Cheek, H. Touati, A. Squicciarini, P.-C. Cheng, User centric policy management in online social networks, in: Proceedings of the 2010 IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY), 2010, pp. 9–13.
[78] I. Kayes, A. Iamnitchi, Aegis: a semantic implementation of privacy as contextual integrity in social ecosystems, in: Proceedings of the Eleventh International Conference on Privacy, Security and Trust (PST), 2013a.
[79] I. Kayes, A. Iamnitchi, Out of the wild: on generating default policies in social ecosystems, in: Proceedings of the IEEE 2013 Workshop on Beyond Social Networks: Collective Awareness, ICC'13, 2013b.
[80] S. Kelly, Identity 'at risk on Facebook', 2008, URL: http://news.bbc.co.uk/2/hi/programmes/click_online/7375772.stm.
[81] E. Mills, Facebook suspends app that permitted peephole, 2008, URL: http://news.cnet.com/8301-10784_3-9977762-7.html.
[82] E. Steel, G.A. Fowler, Facebook in privacy breach, 2010, URL: http://online.wsj.com/article/SB10001424052702304772804575558484075236968.html.
[83] J. Saltzer, M. Schroeder, The protection of information in computer systems, Proc. IEEE 63 (9) (1975) 1278–1308.
[84] P. Hu, R. Yang, Y. Li, W.C. Lau, Application impersonation: problems of OAuth and API design in online social networks, in: Proceedings of the Second Edition of the ACM Conference on Online Social Networks, ACM, 2014, pp. 271–278.
[85] T. Reynaert, W. De Groef, D. Devriese, L. Desmet, F. Piessens, PESAP: a privacy enhanced social application platform, in: Proceedings of the 2012 International Conference on Privacy, Security, Risk and Trust, 2012, pp. 827–833.
[86] D. Devriese, F. Piessens, Noninterference through secure multi-execution, in: Proceedings of the 2010 IEEE Symposium on Security and Privacy, IEEE Computer Society, 2010, pp. 109–124.
[87] A. Besmer, H.R. Lipford, M. Shehab, G. Cheek, Social applications: exploring a more secure framework, in: Proceedings of the Fifth Symposium on Usable Privacy and Security, ACM, 2009, pp. 2:1–2:10.
[88] Y. Cheng, J. Park, R. Sandhu, Preserving user privacy from third-party applications in online social networks, in: Proceedings of the Twenty-second International Conference on World Wide Web Companion, ACM, 2013, pp. 723–728.
[89] S. Kavianpour, Z. Ismail, B. Shanmugam, Classification of third-party applications on Facebook to mitigate users' information leakage, in: World Conference on Information Systems and Technologies, Springer, 2017, pp. 144–154.
[90] M. Egele, A. Moser, C. Kruegel, E. Kirda, PoX: protecting users from malicious Facebook applications, Comput. Commun. 35 (12) (2012) 1507–1515.
[91] K. Singh, S. Bhola, W. Lee, xBook: redesigning privacy control in social networking platforms, in: Proceedings of the Eighteenth Conference on USENIX Security Symposium, USENIX Association, 2009, pp. 249–266.
[92] A. Shakimov, L.P. Cox, MUTT: a watchdog for OSN applications, in: Proceedings of the First ACM SIGOPS Conference on Timely Results in Operating Systems, ACM, 2013, pp. 6:1–6:14.
[93] P. Commissioner, Facebook needs to improve privacy practices, investigation finds, 2009, URL: https://www.priv.gc.ca/media/nr-c/2009/nr-c_090716_e.asp.
[94] M.M. Lucas, N. Borisov, Flybynight: mitigating the privacy risks of social networking, in: Proceedings of the Seventh ACM Workshop on Privacy in the Electronic Society, ACM, 2008, pp. 1–8.
[95] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, D. Starin, Persona: an online social network with user-defined privacy, in: Proceedings of the ACM SIGCOMM Conference on Data Communication, ACM, 2009, pp. 135–146.
[96] S. Guha, K. Tang, P. Francis, NOYB: privacy in online social networks, in: Proceedings of the First Workshop on Online Social Networks, ACM, 2008, pp. 49–54.
[97] W. Luo, Q. Xie, U. Hengartner, FaceCloak: an architecture for user privacy on social networking sites, in: Proceedings of the 2009 International Conference on Computational Science and Engineering, vol. 3, 2009, pp. 26–33.
[98] M. Conti, A. Hasani, B. Crispo, Virtual private social networks and a Facebook implementation, ACM Trans. Web (TWEB) 7 (3) (2013) 14.
[99] A. Smith, 6 new facts about facebook, 2014, URL: http://www.pewresearch.org/fact-tank/2014/02/03/6-new-facts-about-facebook/.
[100] E. Balsa, L. Brandimarte, A. Acquisti, C. Diaz, S. Gurses, Spiny CACTOS: OSN users attitudes and perceptions towards cryptographic access control tools, in: Proceedings of Workshop on Usable Security, Springer-Verlag, 2014.
[101] S. Buchegger, D. Schiöberg, L.-H. Vu, A. Datta, PeerSoN: P2P social networking: early experiences and insights, in: Proceedings of the Second ACM EuroSys Workshop on Social Network Systems, ACM, 2009, pp. 46–52.

[102] L. Cutillo, R. Molva, T. Strufe, Privacy preserving social networking through decentralization, in: Proceedings of the Sixth International Conference on Wireless On-Demand Network Systems and Services, 2009, pp. 145–152.

[103] K. Graffi, C. Gross, D. Stingl, D. Hartung, A. Kovacevic, R. Steinmetz, LifeSocial.KOM: a secure and P2P-based solution for online social networks, in: Proceedings of the 2011 Consumer Communications and Networking Conference (CCNC), IEEE, 2011, pp. 554–558.

[104] L.M. Aiello, G. Ruffo, Lotusnet: tunable privacy for distributed online social network services, Comput. Commun. 35 (1) (2012) 75–88.

[105] L. Aiello, M. Milanesio, G. Ruffo, R. Schifanella, Tempering Kademlia with a robust identity based system, in: Proceedings of the Eighth International Conference on Peer-to-Peer Computing, 2008, pp. 30–39.

[106] A. Shakimov, H. Lim, R. Caceres, L. Cox, K. Li, D. Liu, A. Varshavsky, Vis- a-vis: privacy-preserving online social networking via virtual individual servers, in: Proceedings of the Third International Conference on Communication Systems and Networks (COMSNETS), 2011, pp. 1–10.

[107] N. Kourtellis, J. Finnis, P. Anderson, J. Blackburn, C. Borcea, A. Iamnitchi, Prometheus: user-controlled P2P social data management for socially-aware applications, in: Proceedings of the Eleventh International Middleware Conference, 2010.

[108] N. Kourtellis, J. Blackburn, C. Borcea, A. Iamnitchi, Enabling social applications via decentralized social data management, ACM Trans. Internet Technol. (TOIT) 15 (1) (2015) 1–26. Special Issue on Foundations of Social Computing

[109] T. Bradley, 45,000 Facebook accounts compromised: What to know, 2012, URL: http://bit.ly/TUY3i8.

[110] Facebook, Statement of Rights and Responsibilities, Facebook2015, URL: https://www.facebook.com/legal/terms.

[111] T. Stein, E. Chen, K. Mangla, Facebook immune system, in: Proceedings of the Fourth Workshop on Social Network Systems, ACM, 2011, pp. 8:1–8:8.

[112] C. Wilson, A. Sala, J. Bonneau, R. Zablit, B.Y. Zhao, Don't tread on me: moderating access to OSN data with SpikeStrip, in: Proceedings of the Third Workshop on Online Social Networks, USENIX Association, 2010.

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

[114] G. Jacob, E. Kirda, C. Kruegel, G. Vigna, Pubcrawl: protecting users and businesses from crawlers, in: Proceedings of the Twenty-First USENIX Security Symposium, 2012, pp. 507–522.

[115] S. Wan, Protecting Web Contents Against Persistent Crawlers, College of William and Mary, 2016 Master's thesis.

[116] P. Dandekar, A. Goel, M.P. Wellman, B. Wiedenbeck, Strategic formation of credit networks, in: Proceedings of the Twenty-First International Conference on World Wide Web, ACM, 2012, pp. 559–568.

[117] A. Ghosh, M. Mahdian, D. Reeves, D. Pennock, R. Fugger, Mechanism design on trust networks, in: Proceedings of the 2007 Internet and Network Economics, Springer Berlin Heidelberg, 2007, pp. 257–268.

[118] BBC, Facebook has More Than 83 Million Illegitimate Accounts, BBC2012, URL: http://www.bbc.co.uk/news/technology-19093078.

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

[120] C. Grier, K. Thomas, V. Paxson, M. Zhang, @spam: the underground on 140 characters or less, in: Proceedings of the Seventeenth ACM Conference on Computer and Communications Security, ACM, 2010, pp. 27–37.

[121] A. Nazir, S. Raza, C.-N. Chuah, B. Schipper, Ghostbusting Facebook: detecting and characterizing phantom profiles in online social gaming applications, in: Proceedings of the Third Conference on Online Social Networks, USENIX Association, 2010.

[122] Q. Cao, M. Sirivianos, X. Yang, T. Pregueiro, Aiding the detection of fake accounts in large scale social online services, in: Proceedings of the Ninth USENIX Conference on Networked Systems Design and Implementation, USENIX Association, 2012.

[123] G. Wang, M. Mohanlal, C. Wilson, M.M. Xiao Wang, H. Zheng, B.Y. Zhao, Social turing tests: crowdsourcing Sybil detection, in: Proceedings of the Twentieth Annual Network and Distributed System Security Symposium (NDSS), 2013.

[124] Z. Yang, C. Wilson, X. Wang, T. Gao, B.Y. Zhao, Y. Dai, Uncovering social network Sybils in the wild, in: Proceedings of the ACM SIGCOMM Conference on Internet Measurement Conference, ACM, 2011, pp. 259–268.

[125] N.Z. Gong, M. Frank, P. Mittal, SybilBelief: a semi-supervised learning approach for structure-based Sybil detection, IEEE Trans. Inf. Forensics Secur. 9 (6) (2014) 976–987.

[126] A. Post, V. Shah, A. Mislove, Bazaar: strengthening user reputations in online marketplaces, in: Proceedings of the Eighth USENIX Conference on Networked Systems Design and Implementation, USENIX Association, 2011.

[127] B. Viswanath, M. Mondal, K.P. Gummadi, A. Mislove, A. Post, Canal: scaling social network-based Sybil tolerance schemes, in: Proceedings of the Seventh ACM European Conference on Computer Systems, ACM, 2012, pp. 309–322.

[128] S. Boyd, A. Ghosh, B. Prabhakar, D. Shah, Gossip algorithms: design, analysis and applications, in: Proceedings of the Twenty-Fourth Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, 2005, pp. 1653–1664.

[129] A. Flaxman, Expansion and lack thereof in randomly perturbed graphs, Internet Math. 4 (2–3) (2007) 131–147.

[130] G. Danezis, P. Mittal., Sybilinfer: detecting Sybil nodes using social networks, in: Proceedings of the 2009 Network and Distributed System Security Symposium (NDSS), 2009.

[131] B. Viswanath, A. Post, K.P. Gummadi, A. Mislove, An analysis of social network-based Sybil defenses, in: Proceedings of the 2010 ACM SIGCOMM Conference, ACM, 2010, pp. 363–374.

[132] L. Xu, S. Chainan, H. Takizawa, H. Kobayashi, Resisting Sybil attack by social network and network clustering, in: Proceedings of the Tenth IEEE/IPSJ International Symposium on Applications and the Internet, IEEE Computer Society, 2010, pp. 15–21.

[133] H. Yu, P.B. Gibbons, M. Kaminsky, F. Xiao, SybilLimit: a near-optimal social network defense against Sybil attacks, IEEE/ACM Trans. Netw. 18 (3) (2010) 885–898.

[134] A.N. Langville, C.D. Meyer, Deeper inside PageRank, Internet Math. 1 (2004) 2004.

[135] Y. Boshmaf, D. Logothetis, G. Siganos, J. Lera, J. Lorenzo, M. Ripeanu, K. Beznosov, H. Halawa, Integro: leveraging victim prediction for robust fake account detection in large scale OSNs, Comput. Secur. 61 (2016) 142–168.

[136] U. Brandes, A faster algorithm for betweenness centrality, J. Math. Sociol. 40 (2) (2001) 163–177.

[137] D. Koll, J. Li, J. Stein, X. Fu, On the state of OSN-based Sybil defenses, in: Proceedings of the 2014 IFIP Networking Conference, IEEE, 2014.

[138] M. Motoyama, D. McCoy, K. Levchenko, S. Savage, G.M. Voelker, Dirty jobs: the role of freelance labor in web service abuse, in: Proceedings of the Twentieth USENIX Conference on Security, USENIX Association, 2011.

[139] Y. Boshmaf, I. Muslukhov, K. Beznosov, M. Ripeanu, The socialbot network: when bots socialize for fame and money, in: Proceedings of the Twenty-seventh Annual Computer Security Applications Conference, ACM, 2011, pp. 93–102.

[140] D. Irani, M. Balduzzi, D. Balzarotti, E. Kirda, C. Pu, Reverse social engineering attacks in online social networks, in: Proceedings of the 2011 Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, 2011, pp. 55–74.

[141] Y. Boshmaf, D. Logothetis, G. Siganos, J. Lería, J. Lorenzo, M. Ripeanu, K. Beznosov, Íntegro: leveraging victim prediction for robust fake account detection in OSNs, in: Proceedings of the 2015 Symposium on Network and Distributed Systems Security, NDSS, 2015.

[142] J. Xue, Z. Yang, X. Yang, X. Wang, L. Chen, Y. Dai, VoteTrust: leveraging friend invitation graph to defend against social network Sybils, in: IEEE INFOCOM, IEEE, 2015, pp. 2400–2408.

[143] A. Mohaisen, A. Yun, Y. Kim, Measuring the mixing time of social graphs, in: Proceedings of the Tenth ACM SIGCOMM Conference on Internet Measurement, ACM, 2010, pp. 383–389.

[144] D. Koll, M. Schwarzmaier, J. Li, X.-Y. Li, X. Fu, Thank you for being a friend: an attacker view on online-social-network-based Sybil defenses, in: Proceedings of the Thirty-Seventh IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW), IEEE, 2017, pp. 157–162.

[145] N. Tran, B. Min, J. Li, L. Subramanian, Sybil-resilient online content voting, in: Proceedings of the Sixth Symposium on Networked System Design and Implementation (NSDI, 2009.

[146] N. Chiluka, N. Andrade, J. Pouwelse, H. Sips, Leveraging trust and distrust for Sybil-tolerant voting in online social media, in: Proceedings of the First Workshop on Privacy and Security in Online Social Media, ACM, 2012, pp. 1:1–1:8.

[147] A. Mislove, A. Post, P. Druschel, K.P. Gummadi, Ostra: leveraging trust to thwart unwanted communication, in: Proceedings of the Fifth USENIX Symposium on Networked Systems Design and Implementation, USENIX Association, 2008, pp. 15–30.

[148] D. DeFigueiredo, E. Barr, TrustDavis: a non-exploitable online reputation system, in: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology, IEEE, 2005, pp. 274–283.

[149] D. Quercia, S. Hailes, Sybil attacks against mobile users: friends and foes to the rescue, in: Proceedings of the 2010 INFOCOM, 2010, pp. 1–5.

[150] B. Viswanath, M. Mondal, A. Clement, P. Druschel, K. Gummadi, A. Mislove, A. Post, Exploring the design space of social network-based Sybil defenses, in: Proceedings of the Fourth International Conference on Communication Systems and Networks (COMSNETS), 2012, pp. 1–8.

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

[152] A. Gubichev, S. Bedathur, S. Seufert, G. Weikum, Fast and accurate estimation of shortest paths in large graphs, in: Proceedings of the Nineteenth ACM International Conference on Information and Knowledge Management, ACM, 2010, pp. 499–508.

[153] J. Zhang, R. Zhang, J. Sun, Y. Zhang, C. Zhang, TrueTop: a Sybil-resilient system for user influence measurement on Twitter, IEEE/ACM Trans. Netw. 24 (5) (2016) 2834–2846.

[154] E. Zangerle, G. Specht, Sorry, I was hacked: a classification of compromised twitter accounts, in: Proceedings of the Twenty-Ninth Annual ACM Symposium on Applied Computing, ACM, 2014, pp. 587–593.

[155] Q. Cao, X. Yang, J. Yu, C. Palow, Uncovering large groups of active malicious accounts in online social networks, in: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2014, pp. 477–488.

[156] B. Viswanath, M.A. Bashir, M. Crovella, S. Guha, K.P. Gummadi, B. Krishnamurthy, A. Mislove, Towards detecting anomalous user behavior in online social networks, in: Proceedings of the Twenty-Third USENIX Security Symposium, 2014.

[157] X. Ruan, Z. Wu, H. Wang, S. Jajodia, Profiling online social behaviors for compromised account detection, IEEE Trans. Inf. Forensics Secur. 11 (1) (2016) 176–187.

[158] A. Ntoulas, M. Najork, M. Manasse, D. Fetterly, Detecting spam web pages through content analysis, in: Proceedings of the Fifteenth International Conference on World Wide Web, ACM, 2006, pp. 83–92.
[159] B. Mehta, S. Nangia, M. Gupta, W. Nejdl, Detecting image spam using visual features and near duplicate detection, in: Proceedings of the Seventeenth International Conference on World Wide Web, ACM, 2008, pp. 497–506.
[160] A. Zinman, J. Donath, Is Britney Spears spam, in: Proceedings of the Fourth Conference on Email and Anti-Spam, Mountain View, CA, 2007.
[161] Y.-R. Lin, H. Sundaram, Y. Chi, J. Tatemura, B.L. Tseng, Splog detection using self-similarity analysis on blog temporal dynamics, in: Proceedings of the Third International Workshop on Adversarial Information Retrieval on the Web, ACM, 2007, pp. 1–8.
[162] D. Boyd, J. Heer, Profiles as conversation: networked identity performance on friendster, in: Proceedings of the Thirty-Ninth Annual Hawaii International Conference on System Sciences, vol. 3, 2006.
[163] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G.M. Voelker, V. Paxson, S. Savage, Spamalytics: an empirical analysis of spam marketing conversion, in: Proceedings of the Fifteenth ACM Conference on Computer and Communications Security, ACM, 2008, pp. 3–14.
[164] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, B.Y. Zhao, Detecting and characterizing social spam campaigns, in: Proceedings of the Tenth ACM SIGCOMM Conference on Internet Measurement, ACM, 2010, pp. 35–47.
[165] L. Spitzner, Honeypots Tracking Hackers, first, Addison-Wesley, 2002.
[166] S. Webb, J. Caverlee, C. Pu, 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, 2008.
[167] K. Lee, J. Caverlee, S. Webb, Uncovering social spammers: social honeypots + machine learning, in: Proceedings of the Thirty-Third International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2010, pp. 435–442.
[168] L. Spitzner, The honeynet project: trapping the hackers, IEEE Secur. Privacy 1 (2) (2003) 15–23.
[169] M. Prince, B. Dahl, L. Holloway, A. Keller, E. Langheinrich, Understanding how spammers steal your e-mail address: an analysis of the first six months of data from project honey pot, in: Proceedings of the Second Conference on Email and Anti-Spam, 2005.
[170] C. Kreibich, J. Crowcroft, Honeycomb: creating intrusion detection signatures using honeypots, SIGCOMM Comput. Commun. Rev. 34 (1) (2004) 51–56.
[171] C. Yang, J. Zhang, G. Gu, A taste of tweets: Reverse engineering twitter spammers, in: Proceedings of the Thirtieth Annual Computer Security Applications Conference, ACM, 2014, pp. 86–95.
[172] G. Stringhini, C. Kruegel, G. Vigna, Detecting spammers on social networks, in: Proceedings of the Twenty-Sixth Annual Computer Security Applications Conference, ACM, 2010, pp. 1–9.
[173] F. Benevenuto, T. Rodrigues, J. Almeida, M. Goncalves, V. Almeida, Detecting spammers and content promoters in online video social networks, in: Proceedings of the 2009 INFOCOM Workshops, 2009, pp. 1–2.
[174] I. Kayes, N. Kourtellis, D. Quercia, A. Iamnitchi, F. Bonchi, The social world of content abusers in community question answering, in: Proceedings of the Twenty-Fourth International World Wide Web Conference, ACM, 2015, pp. 570–580.
[175] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, M. Tesconi, The paradigm-shift of social spambots: evidence, theories, and tools for the arms race, in: Proceedings of the Twenty-Sixth International Conference on World Wide Web Companion, International World Wide Web Conferences Steering Committee, 2017, pp. 963–972.
[176] B. Viswanath, M.A. Bashir, M.B. Zafar, S. Bouget, S. Guha, K.P. Gummadi, A. Kate, A. Mislove, Strength in numbers: robust tamper detection in crowd computations, in: Proceedings of the 2015 ACM on Conference on Online Social Networks, ACM, 2015, pp. 113–124.
[177] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, M. Tesconi, Social fingerprinting: detection of spambot groups through dna-inspired behavioral modeling, IEEE Trans. Dependable Secur. Comput. PP (99) (2017) 1.
[178] R. Yu, X. He, Y. Liu, Glad: group anomaly detection in social media analysis, ACM Trans. Knowl. Discov. Data (TKDD) 10 (2) (2015) 18.
[179] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, S. Yang, Catching synchronized behaviors in large networks: a graph mining approach, ACM Trans. Knowl. Discov. Data (TKDD) 10 (4) (2016) 35.
[180] M. Giatsoglou, D. Chatzakou, N. Shah, A. Beutel, C. Faloutsos, A. Vakali, Nd-Sync: detecting synchronized fraud activities, in: Proceedings of the 2015 Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2015, pp. 201–214.
[181] E. Ferrara, O. Varol, C. Davis, F. Menczer, A. Flammini, The rise of social bots, Commun. ACM 59 (7) (2016) 96–104.
[182] Z. Chu, Detecting social spam campaigns on twitter, in: Proceedings of the 2012 Conference on Applied Cryptography and Network Security, Springer Berlin, Heidelberg, 2012, pp. 455–472.
[183] C. Kreibich, C. Kanich, K. Levchenko, B. Enright, G.M. Voelker, V. Paxson, S. Savage, Spamcraft: an inside look at spam campaign orchestration, in: Proceedings of the Second USENIX Workshop on Large-Scale Exploits and Emergent Threats, LEET, 2009.
[184] H. Gao, Y. Yang, K. Bu, Y. Chen, D. Downey, K. Lee, A. Choudhary, Spam ain't as diverse as it seems: Throttling OSN spam with templates underneath, in: Proceedings of the Thirtieth Annual Computer Security Applications Conference, ACSAC '14, ACM, 2014, pp. 76–85.
[185] J. Mirkovic, P. Reiher, A taxonomy of DDoS attack and DDoS defense mechanisms, ACM SIGCOMM Comput. Commun. Rev. 34 (2) (2004) 39–53.
[186] C. McCarthy, Twitter crippled by denial-of-service attack, 2009, URL: http://news.cnet.com/8301-13577_3-10304633-36.html.
[187] E. Athanasopoulos, A. Makridakis, S. Antonatos, D. Antoniades, S. Ioannidis, K.G. Anagnostakis, E.P. Markatos, Antisocial networks: turning a social network into a botnet, in: Proceedings of the Eleventh International Conference on Information Security, Springer, 2008, pp. 146–160.
[188] B.E. Ur, V. Ganapathy, Evaluating attack amplification in online social networks, in: Proceedings of the 2009 Web 2.0 Security and Privacy Workshop, 2009.
[189] H. Gao, J. Hu, T. Huang, J. Wang, Y. Chen, Security issues in online social networks, IEEE Internet Comput. 15 (4) (2011) 56–63.
[190] D.R. Ellis, J.G. Aiken, K.S. Attwood, S.D. Tenaglia, A behavioral approach to worm detection, in: Proceedings of the 2004 ACM Workshop on Rapid Malcode, ACM, 2004, pp. 43–53.
[191] W. Xu, F. Zhang, S. Zhu, Toward worm detection in online social networks, in: Proceedings of the Twenty-Sixth Annual Computer Security Applications Conference, ACM, 2010, pp. 11–20.
[192] I. Kayes, N. Kourtellis, F. Bonchi, A. Iamnitchi, Privacy concerns vs. user behavior in community question answering, in: Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), IEEE, 2015, pp. 681–688.
[193] I. Kayes, Content Abuse and Privacy Concerns in Online Social Networks, University of South Florida, 2015 Ph.D. thesis.
[194] J.R. Mayer, J.C. Mitchell, Third-party web tracking: policy and technology, in: Proceedings of the 2012 IEEE Symposium on Security and Privacy, IEEE, 2012, pp. 413–427.
[195] B. Krishnamurthy, Privacy and online social networks: can colorless green ideas sleep furiously? IEEE Secur. Privacy 11 (3) (2013) 14–20.
[196] Y. Takano, S. Ohta, T. Takahashi, R. Ando, T. Inoue, MindYourPrivacy: design and implementation of a visualization system for third-party web tracking, in: Proceedings of the Twelfth International Conference on Privacy, Security and Trust, IEEE, 2014, pp. 48–56.
[197] M. Gruteser, D. Grunwald, Anonymous usage of location-based services through spatial and temporal cloaking, in: Proceedings of the First International Conference on Mobile Systems, Applications and Services, MobiSys '03, ACM, New York, NY, USA, 2003, pp. 31–42, doi:10.1145/1066116.1189037.
[198] X. Xiao, Y. Tao, Personalized privacy preservation, in: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, ACM, 2006, pp. 229–240.
[199] Z. Zhu, G. Cao, Applaus: a privacy-preserving location proof updating system for location-based services, in: Proceedings of the 2011 IEEE INFOCOM, IEEE, 2011, pp. 1889–1897.
[200] W. He, X. Liu, M. Ren, Location cheating: a security challenge to location-based social network services, in: Proceedings of the Thirty-First International Conference on Distributed Computing Systems (ICDCS), IEEE, 2011, pp. 740–749.

Imrul Kayes (https://imrulkayes.github.io/) is a Senior Data Scientist at Sonobi, Inc., USA. He received the Ph.D. degree in Computer Science and Engineering from the University of South Florida, Tampa, FL, USA, in 2015. He has worked for a number of companies, including Yahoo, Software People, Delta life Insurance, and Binary Solutions. His research interests span the areas of big data analytics, machine learning, and user behavior. His research aims at mining large-scale networks, such as online social, community Q/A, and blogging networks to extract lessons for limiting content abuse, creating user engagement, and defining appropriate privacy.

Adriana (Anda) Iamnitchi is a Professor of Computer Science in the Department of Computer Science and Engineering at University of South Florida. She completed her M.Sc. and Ph.D. in Computer Science at University of Chicago working under the direction of Ian Foster. Before coming to the US for graduate school, she studied at Politehnica University of Bucharest, Romania, from which she received a B.Sc and M.Sc. in Computer Science. Her research interests are in distributed systems, with current emphasis on designing and evaluating socially-aware distributed systems and on characterizing social networks. She is a recipient of the National Science Foundation CAREER Award (2010) and a member of ACM and IEEE.