Computer Science Review 27 (2018) 154–176

Contents lists available at ScienceDirect

Computer Science Review

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

A survey on privacy in decentralized online social networks

Andrea De Salve [a,b,*], Paolo Mori [a], Laura Ricci [b]

[a] Institute of Informatics and Telematics - National Research Council, Via Moruzzi, 1, 56124, Pisa, Italy
[b] Department of Computer Science - University of Pisa, Largo Bruno Pontecorvo, 3, 56127 Pisa, Italy

ARTICLE INFO

Article history:
Received 27 July 2017
Received in revised form 13 January 2018
Accepted 16 January 2018
Available online 3 February 2018

Keywords:
Data privacy
Decentralized online social network
Access control
Security
Peer to peer computing

ABSTRACT

Decentralized Online Social Networks (DOSNs) have recently captured the interest of users because of the more control given to them over their shared contents. Indeed, most of the user privacy issues related to the centralized Online Social Network (OSN) services (such as Facebook or Google+) do not apply in the case of DOSNs because of the absence of the centralized service provider. However, these new architectures have motivated researchers to investigate new privacy solutions that allow DOSN's users to protect their contents by taking into account the decentralized nature of the DOSNs platform.

In this survey, we provide a comprehensive overview of the privacy solutions adopted by currently available DOSNs, and we compare them by exploiting several criteria. After presenting the differences that existing DOSNs present in terms of provided services and architecture, we identify, for each of them, the privacy model used to define the privacy policies and the mechanisms for their management (i.e., initialization and modification of the privacy policy). In addition, we evaluate the overhead introduced by the security mechanisms adopted for privacy policy management and enforcement by discussing their advantages and drawbacks.

Contents

1. Introduction

Recent years have seen unprecedented growth in the Online Social Network (OSN) services [1], with about 300 OSNs collecting

* Corresponding author at: Institute of Informatics and Telematics - National Research CouncilVia Moruzzi, 1, 56124, Pisa, Italy.
E-mail addresses: andrea.desalve@iit.cnr.it (A. De Salve), paolo.mori@iit.cnr.it (P. Mori), ricci@di.unipi.it (L. Ricci).

https://doi.org/10.1016/j.cosrev.2018.01.001
1574-0137/

information about more than half a billion registered users.[1] An OSN enables its users to define their own *profiles*, a virtual representation of themselves, and to explicitly declare the relationships with (the profiles of) other users. Regardless of their purpose, the main service provided by the OSNs to their users is the sharing of information with a set of selected contacts. Users can publish on their profiles very heterogeneous contents, ranging from personal information, wall posts, photos, videos, comments to other posts, and they can send private messages.

Nowadays, the most popular OSNs are based on a centralized architecture where the service provider (e.g., Facebook) acts as central authority and takes control over users' information, by storing a huge amount of private and possibly sensitive information on users and their interactions (such as the personal information and lifestyle behaviors).

Due to the centralized infrastructures, users of the current OSNs are exposed to several privacy risks. Indeed, users of centralized OSNs are forced to share the information directed to their friends by means of the OSN service providers, increasing the risk of censorship, surveillance and information revelation. Indeed, recent events have shown that, in addition to malicious users (internal or external to the OSN), also the centralized service provider [2,3] and third-party applications [4] introduce new privacy risks. The National Security Agency (NSA) documents clearly illustrate how the agencies collected users' information by exploiting the weaknesses of the Facebook's security platform [3].

To address the previous privacy issues and leave to the users the control on their data, researchers have proposed to decentralize the functionalities of OSNs by implementing them in a distributed way. The resulting platforms are known as Decentralized Online Social Networks (DOSNs) [5,6] and they are typically based on a P2P architecture, such as a network of trusted servers, an opportunistic network, a Distributed Hash Table, or an unstructured P2P network. For this reason, in a DOSN there is no central control authority which manages and maintains available the users contents. Instead, DOSNs are based on a set of peers that store the contents and execute the tasks needed to provide a seamless service (such as, search for data [7], recommendation [8], etc.).

For instance, Diaspora [9] is one of the most popular DOSNs which currently has about 669,000 users, and it is based on a network of independent, federated servers that are managed by the users. A federated network is also used by Friendica [10], another popular DOSN based on an extensible plug-in architecture, which currently has more than 1100 users. RetroShare [11], instead, is a DOSN which exploits Friend-to-Friend network to manage and to store users' data.

Therefore, DOSNs allow to shift the control over data to the end users because contents generated by the users are not stored and managed by a single OSN provider but, instead, are the users who have the control over the management of data. However, the decentralization of the service introduces several issues related to the availability of users' contents and their privacy with respect the other users of the system.

1.1. Motivations

While decentralization gives the possibility for increasing the privacy of users with respect to the OSN provider, several studies show that privacy is an increasing concern also for DOSNs' users [12,13]. Indeed, regardless of their architectures, one of the main features provided by current DOSNs is the capability given to the users to define privacy preferences on the contents of their profiles, i.e., to define which other users are allowed to see such contents. In fact, the lack of privacy mechanisms with a suitable granularity level and flexibility could lead to a unwanted disclosure of information, thus exposing users to a number of security risks. Since the number of users' contacts, as well as the number and the type of contents shared on DOSNs, are constantly increasing, members of DOSNs need an effective way to define authorizations to protect their contents.

Users' contents must be protected by the DOSN infrastructure according to users' privacy policies from unauthorized access, i.e., only users who have been granted the proper permissions through privacy policies should be enabled to access the contents. However, while the contents produced by user u may be stored on the devices of u until u is online, when u goes offline these contents must be stored, in order to keep them available, on the devices of other users which are supposed to remain online in the system or on external trusted resources. This requires the usage of proper strategies to prevent unauthorized access to contents of u when they are stored on the devices of other users.

In recent years, several DOSNs have been proposed which try to address these privacy concerns by adopting different approaches. Some previous works have already investigated these DOSNs but they present limitations, for instance because they do not focus on privacy controls and their performance. One of the preliminary works which provides an overview of current DOSN approaches is [6]. However, it only discusses the differences in the design choices for decentralizing the online social network's service without focusing on privacy aspects. In another relevant study, T. Paul et al. [14] proposed a classification of the decentralizing storage mechanisms adopted by current DOSNs and discussed the implications on privacy. The authors of [15] extended the previous work by considering also data integrity and social search issues. In particular, they presented a fine-grained classification of common privacy solution in centralized and decentralized OSNs. Another relevant study, carried out by D. Koll et al. [16], proposed a taxonomy summarizing the approaches adopted by DOSNs in order to solve their major challenges. Behind these, there exist other prior studies reviewing or classifying privacy solutions adopted by current DOSNs [17]. Authors of [18] discuss the benefits that different kind of DOSN architectures have on privacy and their suitability for mobile devices. Authors of [19] only list whether DONSs support user authentication, confidentiality, data integrity, and whether they are resilient to attacks. Even if these works have led to a better understanding of the privacy properties provided by DOSNs, the solutions adopted by current DOSNs to implement their privacy controls deserve a more detailed study. Indeed, previous works do not take into account the types of privacy policies provided the current DOSNs. In addition, they do not consider the performance of privacy controls, i.e., they do not investigate in detail the overhead introduced by the mechanism adopted to enforce the privacy policies. In many cases, the performance of privacy mechanisms for protecting contents is crucial for the success of the service. For instance, the designers of some DOSNs (such as LifeSocial.KOM [20] and PeerSoN [21]) investigated the overhead introduced by encryption schemes used by their DOSNs and highlighted the impact they have on performance and user experience [22,23]. In addition, the choice of the cryptographic schemes does not only involve performance aspects, but also privacy aspects related to the capability provided by the DOSNs to define different types of privacy policies and to modify them. As a result, some important aspects of the privacy solutions of the DOSNs deserve further study and there is the need to investigate more deeply several crucial characteristics related to these solutions in order to allow a better design of the next-generation DOSN infrastructures.

[1] See: http://en.wikipedia.org/wiki/List_of_social_networking_websites.

1.2. Contributions

The main aim of this paper is the investigation of the different approaches used by existing DOSNs to protect the privacy of the contents of their users. For this reason, we identified a large set of DOSNs which have been proposed in the literature, by considering both the DOSNs which are really deployed (such as Diaspora, Friendica, or RetroShare) and the ones which are under active development. For each of the selected DOSN, at first we briefly analyze the architectural model used to provide independence from a centralized provider, and then we study the approach adopted to enable users to define their privacy preferences and to enforce them, i.e., the *privacy model* and the *privacy policy management*. In particular, the privacy model is related to the capability given to the users to define which other users are authorized to access their contents. To help the reader in understanding the characteristics of the different privacy models, we defined a taxonomy to classify them. The policy management model, instead, is related to the solutions adopted to guarantee that the privacy policies defined by the users are actually enforced on their contents. In addition, we also investigate the approaches used to enable the modification of the privacy preferences on the contents, i.e., to grant access to new users or to revoke access to previously authorized users. Since most DOSNs exploit cryptographic techniques, we evaluate the overhead introduced by the privacy policy management in terms of number of cryptographic keys created, and number of encryption operations required. For each DOSN, we also determine whether and how it ensures (or not) the backward secrecy property [24] when the privacy policy is changed. Besides presenting a comprehensive comparison of the privacy mechanisms used by the current DOSNs, this paper enables the reader to better figure out the weakness as well as the strength provided by privacy mechanisms of the current DOSNs.

1.3. Outline of the survey

This paper is structured as follows. We provide a description of the architectures and design choices of the selected set of DOSNs in Section 2. In Section 3 we identify the main requirements concerning privacy of the contents. Section 4 summarizes the privacy models proposed by current DOSNs while Section 5 describes the mechanisms used for privacy policy management (i.e., initialization and modification of a privacy policy). Finally, we evaluate the privacy provided by current DOSNs in Section 6 and discuss their limitations and advantages in Section 7. Finally, we draw the conclusion and final remarks in Section 8.

2. Decentralizing the Online Social Networks

A current trend for developing OSNs that do not rely on a centralized service provider is moving towards the decentralization of the OSN service. A Decentralized Online Social Network [6] is a OSN implemented in a distributed and decentralized way. The approaches exploited by current DOSNs to provide independence from a centralized provider are typically based on Peer to Peer (P2P) architectures (such as a Distributed Hash Table [25] or network of interconnected trusted servers). Indeed, every participating user can act both as a server and as a client, depending on the context [26]. The approaches used by current DOSNs to provide independence from a centralized authority combine multiple architectural levels, each with its own features. According to the topology of the P2P network, the currently available DOSNs can be classified into two alternative P2P architectural styles:

Structured: In structured P2P architectures, the peers are organized into a specific topology that ensures good performance on specific tasks of the system, such as routing. This architecture exploits hashing to associate an identifier to the peer and to pair contents with peers, so defining a DHT.

Unstructured: This P2P architecture does not impose any particular structure and resources are connected according to their needs. Operations are usually implemented by using flooding or gossip-like communication between users.

Instead, the approaches used by current DOSNs to accomplish the data storage functionality are mainly based on three P2P architectural styles:

Decentralized: This architecture does not impose any particular conditions concerning where data should be stored, since contents of users are stored on random nodes.

Semi-decentralized: A subset of the users in the system (super peers) take responsibility for storing and managing information of all the users. The choice of providing super peer services can be voluntary or incentive-based.

Hybrid: This architecture exploits the P2P approach, but also relies on some external service provided by a centralized entity (such as Clouds, Private Servers, Dropbox, etc.). This service allows the users to exploit permanently available resources which guarantee that their contents can be always accessed, but this also implies a cost for the DOSN's users.

For a comprehensive description of the DOSNs structure we refer to [16], a survey which is focused on the architectural style of the main DOSNs. In the following, we provide a list of the currently available DOSNs which will be investigated in this paper, along with a summary of their architectural characteristics. Indeed, the architectural styles used by current DOSNs to provide their services can impact the privacy mechanisms, as well as the privacy of users.

The architectural style adopted by DOSNs is mainly based on structured P2P architectures, such as OpenDHT in PeerSoN, FreePastry in LifeSocial.KOM, Likir in LotusNet, DECENT, Cachet, SocialGate, and eXO. Indeed, these structured P2P architectures have proven to be reliable solutions to deal with the dynamism of peers (churn) and with load balancing. Structured P2P architectures are very efficient for routing information based on a key and they are exploited also to implement anonymous communications or to get updated status information about a peer (such as, IP address, online status, ports, etc.). In addition, structured P2P architectures have proven to be very efficient in retrieving information managed by the peers and several DOSNs exploit this advantage by storing users data on the peers of a structured P2P network. As for instance, PeerSoN, LotusNet, LifeSocial.KOM, Cachet, DECENT, and eXo exploit a DHT to store and to replicate encrypted contents of the users on the peers of the DHT. For this reason, data are typically stored encrypted to prevent the owner from accessing them and replicated on different peers to increase their availability. However, this solution has a limitation in case of relational data, such as those generated by the DOSN users, which are typically organized in logically connected structures (for example a post with its comments and likes). Indeed, the DHT is not able to deal efficiently with relational data because it needs many accesses in order to retrieve the complete data structure, taking up to hundreds of second [27]. For example, in order to obtain the complete data structure concerning a post on the profile of a user u, the applicants have to access the profile of u, retrieve the post, retrieve the comments linked to the post and the likes related to both post and comments. For this reason, the most part of the existing DOSN systems exploit structured overlay networks to store a reference to the peers having the user's contents. As for instance, ProofBook

replicates encrypted contents on the peers of the users entitled to access these contents and it exploits Proof of Work in order to mitigate DoS attacks towards the replica peers. Instead, SocialGate requires user to deploy a gateway node which acts as router and ensure higher data availability.

Behind these, there are also DOSNs (such as Diaspora, Friendica, SuperNova, Persona, Vis-a-Vis and Vegas) that rely exclusively on unstructured P2P architecture where all the users' devices (or a subset of them) can act as super-peers by providing different types of services. This solution mitigates the overhead needed for a peer to connect to the system because the absence of structure reduces both complexity and prone to dynamism. As for instance, in Diaspora, Friendica, and SuperNova some users can decide to take part of the federated network by acting as pods or Friendica servers. In general, these DOSNs can provide either a semi-decentralized or a hybrid storage service. In the first case, the contents of users are stored on a sub-set of peers provided by users of the DOSN (such as Diaspora, Supernova, or Friendica) while in the second case contents are stored by third parties (such as Vis-a-Vis, Persona, Contrail or Vegas). Depending on the assumptions made by each DOSNs, contents can be stored either unencrypted (such as, Diaspora, Vis-a-Vis, and Friendica) or encrypted. In the former case, the DOSN requires the user to trust the peers that have been chosen to store unencrypted contents. In addition, DOSNs based on semi-decentralized data storage may or may not ensure replication of the data on different peers for availability purpose. As for instance, Diaspora, SocialGate, and Friendica, do not provide data replication because they assume that peers chosen by users to store their contents are very reliable. In fact, run a Diaspora pod requires a lot of memory because the database grows very fast. In addition, SocialGate enables users to select a mirror server in the case of the home gateway is not reachable. In addition, computational and network resources required by the server depend on the number of users hosted by the pod and how much traffic the pod receives from other pods. In case of a high number of data lookups the robustness of semi-structured P2P system is heavily affected due to the network congestion caused by numerous queries. Another example is Friendica server, that requires to run PHP/MySQL/Apache and other components to be installed.

Contrail, Vis-a-Vis, Persona, and Vegas, leverage external storage system to ensure better scalability, performance and availability of data. Indeed, availability of contents is guaranteed by exploiting external centralized data storage services, such as FTP, WebDAV, Amazon S3, Google Drive, or Dropbox in Vegas and Persona or Virtual Identification Servers in Vis-a-Vis, Azure Cloud in Contrail.

Finally, many DOSNs which in the past relied completely on structured or on unstructured P2P architectures, have been redesigned to exploit a hybrid architecture that takes advantage of both solutions. In particular, Safebook, My3, DiDuSoNet, Prometheus, Gemstone, Soup and RetroShare have enhanced their platforms by integrating both structured and unstructured overlays. In most of the proposed solutions, the structured P2P level is used to find friends (e.g., RetroShare) or to find the peers where data are stored (such as Safebook, My3, DiDuSoNet, Prometheus, Soup, SocialGate, and Gemstone). The unstructured P2P level provides a semi-decentralized storage service which consists of the peers selected by users to store their data and it is used to retrieve the data from the corresponding peers. Instead, the structured P2P level is used as an index to speedup the lookup of data and for the routing. The most part of the DOSNs considered in this category rely on the replication of the contents in order to increase data availability, while contents can be stored either encrypted on different peers (such as Safebook, Prometheus, Soup and Gemstone) or unencrypted on the peers of trusted friends. In the first case, the DOSNs store user's contents on any peers of the system while in the second case contents are stored on a subset of the users (super peers) in the system. However, the choice of the super peers where to store replicas of the contents is typically demanded to the users while only few DOSNs (such as Soup, DiDuSoNet, and [28–30]) adopt trust models to automatically derive these super peers. Table 1 classifies DOSNs according to their storage mechanisms and specifies if the corresponding DOSN uses encryption (Enc) or replication (Rep) of the contents, as well as the main characteristics of the privacy solutions exploited by current DOSNs (which will be discussed in the next section).

3. Privacy requirements in DOSNs

Decentralized OSNs address the main privacy concern about users' data that affects centralized OSNs, because data are stored on the peers of the users belonging to the DOSN or on some storage server chosen by the user, and there is no central authority that controls and stores such data. In addition, DOSNs users are able to define privacy policies, i.e., (typically simple) statements specifying who can access their contents. As a result, DOSNs shift the control over users' data to the peers that build up the system (i.e., to the users these peers belong to), thus solving some, but introducing new security issues, such as the one concerning the confidentiality of users' data with respect to the users providing the peers where such data are stored. In particular, DOSNs ensure data availability by allocating contents on the peers of users who may not be authorized to access them according to the privacy policies defined by the content owners. Consequently, the adoption of a proper security support to protect the privacy of such contents is required. Fig. 1 shows a graphic representation of the privacy policy defined by a user U that defines the contents $C = \{c_1, \ldots, c_k\}$ that each user belonging to the set $A = \{u_1, \ldots, u_l\}$ can access. Since no centralized storage service exists in DOSNs, contents $c_1, \ldots, c_k$ of U can be stored on the device of a user $p \notin A$ (i.e., a user that does not belong to the set of authorized users). Hence, privacy in DOSNs is guaranteed by allowing users to express their preferences to decide which information should be disclosed to the other users, while proper security mechanisms are exploited to protect the confidentiality of these contents in order to disclose them according to the privacy preferences previously defined. Based on the above considerations, we can identify the following main features concerning privacy in DOSNs:

Privacy model: Privacy model is defined as the capability of the DOSN to provide different types of privacy policies enabling the users to specify the set of members who are entitled to access their contents. Indeed, the way in which the users can express the privacy preferences concerning their contents, on the one hand, heavily impacts on the capability of the access control mechanism and, on the other hand, increases awareness about the audience accessing the contents. Typically, privacy policies are simple statements specifying who has access to the user's contents in terms of a set of OSN features (such as friendship type, interests, work, school,…).

Privacy policy management: Once users have defined privacy policies on their contents, the DOSNs framework must guarantee that these policies are enforced on each content by using proper security mechanisms. The enforcement of privacy policies ensures that user decisions are properly implemented and the related contents are disclosed only to authorized people. In addition, users are able to change their privacy policies by adding new users or removing old users from the set of authorized ones.

Table 1
Comparison of the security mechanisms provided by current DOSNs.

Storage	DOSN	Enc	Rep	Schema	Privacy policy
Decentralized	**DECENT**	✓	✓	Asymmetric, Symmetric, ABE	selected contacts, attribute-based groups
	LotusNet	✓	✓	Asymmetric, Symmetric	selected contacts, regular expression on content type
	LifeSocial.KOM	✓	✓	Asymmetric, Symmetric	private, public, selected contacts
	Cachet	✓	✓	Asymmetric, Symmetric, ABE	identity or attribute-based policy
	eXO	–	✓		public, private
	RetroShare	✗	✗		circles, selected groups, selected contacts, n-degree contacts
	PeerSoN	✓	✓	Asymmetric, Symmetric	private, public, groups
Semi Decentralized	**Gemstone**	✓	✓	Symmetric, ABE	attribute-based policy
	Safebook	✓	✓	Asymmetric, Symmetric	private, group, attributes, trust level, depth
	SuperNova	✓	✓	Asymmetric, Symmetric	private, public, selected contacts
	ProofBook	✓	✓	Asymmetric, Symmetric	selected contacts, group
	Soup	✓	✓	ABE	attribute-based policy
	Prometheus	✓	✓	Asymmetric	relationship type, interactions, weights of the relationship, location
	DiDuSoNet	✗	✓		selected contacts, all contacts, Dunbar circles
	My3	✗	✓		trusted contacts, all friends
	Diaspora	✗	✗		private, public, selected contacts
	Friendica	✗	✗		public, selected groups, selected contacts
Hybrid	**Persona**	✓	✓	Asymmetric, Symmetric, ABE	private, group, selected contacts, attribute-based group
	SocialGate	✓	✗	Symmetric, ABE	selected contacts, attribute-based groups
	Vegas	✓	✗	Asymmetric	selected contacts, all friends
	Contrail	✓	✗	Asymmetric, Symmetric	white-list (ACL),filter based on users' identities, location, tags or keywords of contents
	Vis-a-Vis	✗	✗		group admission based on friendship and credentials

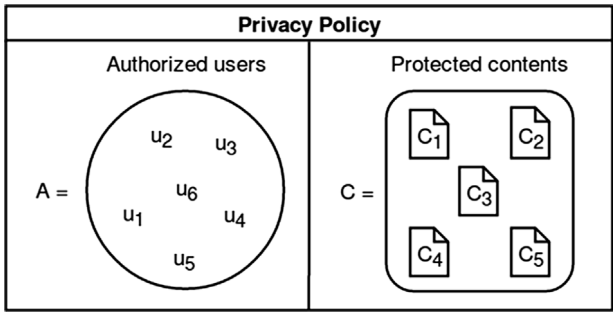

Fig. 1. The elements of a privacy policy.

4. Privacy model

Each DOSN enables its users to protect their contents by defining privacy policies that determine the set of users authorized to access each of them. The majority of existing DOSNs, provide to the users a limited and predefined set of privacy policies based on the knowledge derived from the social network, e.g., relationships (friends, family, colleagues, etc.), groups, content or profile information. For instance, some DOSNs allow their users to define groups of friends, and to specify which groups are allowed to access each of the content they publish. Table 1 summarizes the access control options of current DOSNs by reporting the privacy policy type and (if the case) the encryption schemes used by each DOSN to enforce privacy policies. The most part of current DOSNs protect users' contents by employing both asymmetric and symmetric encryption. The details about the encryption schemes used to enforce privacy policies will be discussed in Section 5. In the following of this section, instead, we give a short description of the privacy model supported by each of the DOSNs introduced in Section 2.

Diaspora. In Diaspora [9], the users define privacy policies based on "aspects", i.e. groups of contacts which are part of one or more aspects of the users' life. Indeed, the "aspects" can be defined to reflect common features of friends (such as common interests, type of the relationships, etc.). The groups are visible only to their owners in their profiles, but the group owner can decide whether to make the identity of the group's members visible to each other (by creating public group) or visible only to the group owner (by creating private group). The aspects mechanism can be used only by the group owner to control the sharing of the contents with the group members. At the moment of content creation, the content owner may decide that the content is public, i.e. visible by everyone, or may select the aspects with which he wants to share the content.

Safebook. The privacy model of Safebook is sketched in [31] and refined in [32]. Personal information of users is organized into atomic attributes, and privacy policies based on these attributes can be defined by each user. Contents (or artifacts) are logically grouped by labels (such as Comments, Posts, Images, etc.) and on each label a set of attributes is defined in order to be exploited in privacy policies. In order to protect their contents, users can define privacy policies based on the type (or label) of the relationship (such as Family, Close friends, etc.), the depth (such as Friends, Friend of a Friend, etc.) or the trust level of the relationship (i.e, a numeric value that user assigns to each friendship relation in order to indicate the level of confidence with the corresponding friend). For these reasons, users can assign labels or attributes to their relationships in order to define badges, i.e., set of contacts having the same label and attributes. Users can also define their custom groups of users, by choosing which of their contacts belong to them (regardless of the relationship type). The group is visible to any members, who are aware of the other users participating in the same group. In addition, Safebook permits the creation of private groups (named *circles*) and, in that case, membership information are visible only to the owner of the group. Contents can be private, i.e., they are not published, protected, i.e. they are published encrypted on mirrors, or can be public, i.e., hey are published on mirror nodes without encryption.

PeerSoN. PeerSoN [33] exploits the concept of "shared space" to abstract social entities like groups, friend networks, or personal profiles. Each shared space is a container for a set of data objects like photo, albums, videos, and may be paired with a set of members. PeerSoN allows its members to define simple access policies based on individual user or private group of users. In particular, the user is able to create a *filegroup*, which is a set of objects having the same authorized readers. Group name remains visible only to the owner of the group and the members of the same filegroup cannot see each other.

LotusNet. In LotusNet [34], access control is achieved using signed grant certificates. The contents created by users are grouped using content type (such as Status, Photos, Comments, etc.) and a grant certificate is generated for each friend. A grant certificate consists of the identities of the owner and of the granted user, an expiration time and a regular expression that is a compressed list of all the content types that can be accessed by the granted user. Grants are paired with social contacts, rather than with shared resources. This strategy allows LotusNet to limit the number of grant certificates to the number of users, but does not allow the definition of granular privacy policies for specific content. In addition, a grant certificate created by user A for a user B implies that a social tie has established between A and B. Through this mechanism, it is possible to represent both directed and undirected social networks, depending on whether grants are reciprocated or not.

SuperNova. SuperNova [35] defines three privacy levels to be paired to each content: public, protected, or private. Public contents can be accessed by any user, while private contents are accessible only by the publisher itself, for example for data backup. Finally, protected contents are intended to be shared with a subset of friends explicitly selected by the content owner. Authorized users cannot see who is allowed to access the same content, but they have information on all the peers, i.e. the storekeepers or the super-peers, which store a replica of the profile of the user. Furthermore, each storekeeper has information about all the other storekeepers which are storing the node's data.

LifeSocial.KOM. LifeSocial.KOM [20] does not allow users to define complex privacy policies, but it provides a security layer [23] where users are enabled to define Access Control Lists (ACLs) [36] containing the identities of friends authorized to access a specific content. In particular, when a content is created, the user creates a privacy policy for that content by selecting the identities of the users authorized to access it. In addition, users are able to create public groups based on a common interest where both group name and the identities of the group members are visible to anyone. At the group level, privacy control requires that contents published within the group are visible only to all the group members.

Vis-a-Vis. The main goal of Vis-a-Vis [37] is the sharing of geographic locations within large social groups, however the framework can be exploited for sharing other social contents. Vis-a-Vis allows its users to create privacy policies to restrict the sharing of locations, based on public or private groups. At the moment of the group creation, the group owner selects an admission policy for the group defining the set of credentials corresponding to the members authorized to access the group. Each user within the group possesses a shared attribute. The credential set can be: (i) empty, in the case of a public group with contents accessible to everyone; (ii) shared attributed such as an inter-personal relationship with the group owner (e.g., family, colleague, or classmate) or an interest in a particular topic [37]. Each user of a group is associated to a geographical region (e.g., hometown or current GPS coordinates) visible to all members of the group. Geographical regions of users are organized according to a hierarchical tree structure where the higher levels of the tree represent coarse-grained areas (such as countries, followed by states, counties) while lower levels of the tree represent fine-grained regions (such as cities and places). When a user u joins a group, u provides a set of geographic regions that specify the geographic information that u want to share with other group members.

My3. Members of My3 [38] leverage their trusted friends to enforce privacy policies on the access requests. However, authors do not specify the exact organization of the profile content and the type of privacy policies that members can define on it.

Cachet. Cachet [27] allows its users to define two kinds of privacy policies: identity-based and attribute-based (AB) policies. Identity-based policies define accesses based on the identities of users. AB policies, instead, are defined through logic formulas over attributes and they are used to define access for a group of social contacts sharing some common features. As for example, an AB policy can grant access to users having attributes *friend*, *coworker*, *family*. In particular, only the type and depth of relationships are used as attributes of the AB policies. Each content may be protected with three policies: a Read Policy which specifies the set of users that can read the content, a Write Policy which is generally set to the identity of the content owner, and an Append Policy which may define, for instance, who can comment on post.

Persona. In Persona [39], users exploit attributes to model semantic properties of their social contacts and to define privacy policies based on them. The authors propose to use as an attribute the type of the relationships among users (such as co-worker, friends, friends of friends, etc.). In addition, users can organize their contacts into private groups which are intended to be used by the group owner. Indeed, groups and group memberships are visible only to the group owner. This makes group management in Persona different from classical scenarios because the members of a group may not necessarily be aware of each other. For instance, Alice may post a message on Bob's wall, encrypted for Bob's friends, without necessarily knowing the list of Bob's friends. Groups can be defined by selecting the identities of users to be added and they are heterogeneous in terms of the type of relationships. The content owner chooses whether to share the content in a private group or by using the relationship type attribute.

eXO. In eXO [40], each user is able to mark a content only as public or private. In the former case, the content can be seen by anyone in the system because it is indexed by the DHT, while private contents are visible only to users having a friendship relation with the content's owner. Similarly, the user's profile may be public, i.e., indexed and replicated by the DHT, or private, that is stored only on the peer of the profile owner. In addition, a content can be paired to a set of tags (i.e., terms that describe the content). However, tags on contents cannot be exploited during the definition of the privacy policies.

Vegas. Members of Vegas [41] have few capabilities in terms of privacy preference settings and they can exploit only a limited set of privacy policies that allow them to share contents only with the selected users or with all friends. When users create a content, they can choose whether to share it with to all users having a friendship relation with them. Alternatively, the content owner individually selects the friends with whom to share the content.

DiDuSoNet. DiDuSoNet [42] defines simple privacy policies that are based on friendship relation. In addition, each user has a profile which contains only public and private data. Public data can be accessed by anyone while private data can be accessed only by contacts having a friendship relation with the content owner.

Prometheus. Prometheus allows users to specify privacy policies based on types of relationship, the labels associated to interactions (such as Post, Comment, etc.), weights related to the trust of the relationship, and locations specified by the users in their profiles. Access Control Policies define white-lists, i.e. list of users who can access the contents, but the user can also specify a black-list to exclude some users. The information referred in the access control policy may be explicitly derived from multiple external sources provided by users, such as e-mail, blog, phone, or other DOSNs. As for example, a privacy policy on trusted peers of a user u can allow both LinkedIn friends and LinkedIn friend of friend to access the work-related information of u. Labels play a fundamental role

in the Access Control Policy definition. For instance, users and contents may be grouped under labels to define proper policies. For instance, a user may group all contents related to work under the "work" label and restrict access to those contents only to users characterized by the "co-worker" label, possibly excluding some single user through the black list. The policies are stored on the trusted peers of the users and they are evaluated when an access request is received.

Gemstone. Users of Gemstone [43] are able to specify privacy policies based on distinct attributes (or properties) derived from the OSN knowledge. In particular, the type of the relationship (i.e., the label associated to the relationship itself) and profile information of the users can be exploited to define privacy policies on contents. The typical profile information used as attributes of privacy policies is location and interest.

Friendica. In Friendica [10] each user is paired to a default public profile which can be access by all the users. Users can restrict the access on their profiles to the intended audience (based on, friends, protocols, email addresses, and DNS location). In addition, a user can maintain different personalities by creating distinct profiles, each configured for the intended audience.

The users are able to organize their contacts in different groups which can be used to restrict the access to the shared contents. In addition, users can select multiple groups and specific friends that are authorized/unauthorized to access the content. A visual editor helps users to manage the members of groups, as well as, to select the friends who can view a specific profile. In addition, Friendica supports also the creation of (one-way) relationships (i.e., follower, fan, etc.)

RetroShare. Users of RetroShare are able to create groups of friends having different permissions. In particular, RetroShare imposes that a friend cannot belong to two different groups of a user, i.e., groups constitute a partition of the peers.[2] In addition, privacy policies for a single friend, all friends or friends of friends are also provided. The privacy policies are defined by the content owner by using flags provided by the visual group interface and each content shared by a user can belong to zero or multiple groups. Each content can be published by using three different sharing option flags: (i) the green flag indicates that only friends in selected groups can see and download the content, (ii) the blue flag enables all the friends to see and to download the content, and (iii) the *N* flag indicates that friends, but also friends of friends, and friends who are at a maximum distance of *N* can download files.

In addition, a user can control the search visibility and anonymous access to the shared contents. As for example, users can define a privacy policy that allows the sharing of a content anonymously with untrusted friends while it denies access to the members of the family group. The group owner selects a discovery policy for each group that specifies the group setting's visibility between the members of the group. In particular, the group owner can decide to allow/disallow: (i) sending information between friends of this group; and (ii) sending information from peers of this group to others.

Finally, in the latest version of RetroShare [44] users are able to create *circles*, i.e., groups of (anonymous) identities, that can be used to restrict the visibility to forums, channels, etc. A circle can be: (i) Public: if members and contents are visible to any friends; (ii) Private: if it is visible only to the members of the circle; or (iii) Restricted: the members and contents are visible only by members of another circle, and (iv) Invited: visible to invited members who explicitly join the circle.

[2] http://retroshare.sourceforge.net/wiki/index.php/Groups.

Contrail. Users are able to define their privacy policies by exploiting *sender-side filters* and *white-list*. In particular, a user can specify a specific filter for each friend that accepts a content produced by the friend as input and returns the authorization decision (true or false). A content consists of a payload and application-defined metadata that can be exploited by filters (such as GPS coordinates, tags, or keyword on contents). The filter owner sent a filter installation request to the proper friend's device. A white-list is defined by each user in order to limit the installation of the filters to specific users' devices. As a result, the installation request only reaches the friend's device if the filter owner has been included in the white-list of the friend. Once the friend accepted the filter installation the contents produced by on his device evaluated by the filter and, if the case, the new content is transmitted to the device of the filter owner.

Soup. To ensure privacy of the contents users of Soup [45] are able to define privacy policy based on attributes, which are mainly related to the user's profile and the relationships. Indeed, users can assign to the relationships defined with their friends attributes that specify the type of relationship (e.g., colleague) or properties related to the friend's profile (e.g., lives in my city).

ProofBook. Users of ProofBook [46] are able to define privacy policy based on either groups or friendships information. Indeed, users can create private groups which reflect the different privacy requirements. As for instance, based on their needs, users of ProofBook can define group of close friends, colleagues, or family. When a user establishes a friendship relationship with another user, he can decide to add this friend to one or several groups, depending on his needs. In addition, the user can decide to keep out the new friend from his private groups and to share some contents only with a specific friend.

DECENT. The user's profile of DECENT [47] is organized by exploiting hierarchical structure where the root object contains references to other objects, such as albums, wall, friends, etc. Each content is stored in a container object that consists of two main component: the main content and a list of comments modeled as references to other container objects. At the time of content creation, the content owner defines three different privacy policies: (i) the read policy specifies who may read the contents, (ii) the write policy describes who may modify the contents, while (iii) the append policy specifies who may comment the contents. As a result, the permission defined by user on the comments can be more restrictive than the permissions on the corresponding content. The policy could be: attribute-based, where a formula over attributes specifies the set of authorized users, identity-based, where identities of users are exploited, or a combination of both types. Each user acts as key authority for their friends by assigning arbitrary set of attributes to his friends.

SocialGate. The users' of SocialGate are able to define privacy policies based on different attributes. In particular, the policy could exploit formula over friendship's attributes to specify the set of authorized users [48]. In addition, attributes can also refer to the identities of users. Each user acts as key authority for their friends by assigning arbitrary set of attributes to his friends.

4.1. Advanced privacy policy mechanisms

Besides the DOSNs previously described, there is a large collection of works that propose extensions to the existing approaches.

Authors in [49] propose the D-FOAF system: a Friend of Friend ontology-based distributed identity management system for DOSNs, where access control management is provided as additional services. In D-FOAF, relationships are paired with a trust level, and users define their access control policies in terms of

minimum trust level and maximum length of the paths (in terms of friendship relationships) connecting the applicant to the content owner. Authors in [50] extend the D-FOAF system by considering the case of multiple types of relationships.

On the same line of research, the authors of [51] propose Lockr: a system exploiting relationships among users within the DOSN to specify privacy policy.

Authors in [52] propose a privacy mechanism based on trust where each user has a reputation value computed by considering the ratings specified by other users in the system. In particular, each user is paired to an operating trust level that is used to determined contents that can be accessed by the user. The operating trust level is obtained by combining an input parameter provided by the user and the reputation value of the user. The content created by a user is paired to numeric confidence level which ranges from 0 (for contents with higher exposure) to the operating trust level of the content owner (for contents with limited exposure). Each content created by a user is encrypted with a key K_c and published on a set of trusted peers. Threshold based cryptography is used as sharing scheme between the trusted peers. The user operating at trust level τ can access the content c to the trusted peers only if the confidence level of the contents is equal or less than the operating trust level τ of the applicant.

Authors of [53,54] focused on a rule-based access control mechanism for OSNs where authorized users are denoted in terms of the type of the relationship, the depth of the paths between two users in term of friendship relations and the trust level of the existing relationship.

Recently, Carminati et al. [55] proposed an access control model based on semantic web technologies where semantic web ontologies are used to model different aspects of the online social network (relationship, properties of the users, relationship between users and resources, etc.).

Authors of [56,30] proposed to exploit XACML [57] (a language based on XML defined by the OASIS consortium) for defining complex privacy policies that leverage the knowledge provided by the DOSN (e.g., time, type of relationship, location, etc.). In addition, authors of [30] propose to exploit such privacy policies to produce smart contents allocation that meets the privacy preferences defined by users.

Typically, the systems reviewed above, exploit privacy policy languages for representing their policies. Privacy policy languages are designed to define the privacy controls that both organizations and users want to express. Privacy policy languages are expected to be fairly simple.

Instead, the authors of [58] focused on the resolution of the privacy conflicts arising from the process of data sharing. In particular, users are able to specify their privacy policies to grant data access to the other users, based on their friendship relation, group membership and identity. Each user is paired to a *trust level* while each privacy policy for a content is paired to a *sensity level*, which are both of numerical values defined by the user who specifies the policy. The trust level indicates how much user trust another member while the sensity level specifies the degree of protection of the data, respectively. The resolution of a privacy conflict aims to find an authorization decision (permit, deny) which ensure lower privacy risk and lower sharing looseness. In particular, authorization decision is computed as a function of the trust level and the sensity level of the data, and the trust level of the applicant.

5. Privacy policy management

In order to enforce privacy policies, the majority of the solutions proposed by current DOSNs are based on encryption mechanisms. Other DOSNs [42,59,29], instead, exploit alternative approaches in order to avoid the use of cryptography.

In the case of cryptography-based DOSNs, encryption mechanisms perform a data transformation in such a way that only authorized users can understand the contents. For instance, to achieve fine-grained access control, each content should be encrypted before being stored on the peers of the DOSN. In turn, the secret key used to secure this content should be securely distributed to the users who are authorized to access the contents (see Sections 5.1 and 5.2). Consequently, even though a generic user can retrieve the encrypted content stored on a peer, only users who have the permission of the owner (i.e. the secret key) can understand it. As a result, cryptographic mechanisms used for privacy policy management introduce some overhead in terms of: number of keys created and number of encryption operations.

Every time a user defines a privacy policy $P(A, C)$ to protect the contents in C, the DOSN must initialize it by generating the encryption data structure, e.g., the cryptographic keys, required to protect these contents, by distribute it among the proper set of user, and by encrypting these contents before being stored on the peers of the system. In addition, every time a user changes a privacy policy, the related encryption structures meant to enforce such policy must be properly updated as well to reflect the new access rights, i.e., to update the set of users allowed to access the related contents. For instance, if the privacy policy model is based on the definition of groups of users, the initialization of a policy concerns the creation of the group key and the distribution of this key to the group members. Every time the privacy policy is changed by adding a new member to the group, the DOSN must properly update the group key and redistribute it to the group members in order to ensure that both the new member and the previously authorized users can access future contents that will be published on this group. This is clearly a performance issue, especially when the set of authorized users specified in a privacy policy is large and it is frequently updated.

The cryptographic systems used by the existing DOSNs are typically based on the combination of symmetric/asymmetric cryptography or their variations (such as Attribute Based Encryption or ABE [60]). In contrast to traditional public–private schemes, in ABE, a set of descriptive attributes is used as an identity to generate a secret key and to encrypt the data. Only the users who holds a secret key with the specified attributes are able to decrypt the data.

Table 2 summarizes the general notation used to represent the key factors affecting the performance and the complexity of a secure DOSNs. In particular, we consider the overhead introduced by each DOSN for the enforcement of a general privacy policy $P(A, C)$ which grants to the set of authorized users $A = \{a_1, \ldots, a_n\}$ the permission to access the set of protected contents $C = \{c_1, \ldots, c_m\}$. Based on the previous analysis, we identified two different operations that can occur during the life time of privacy policies: *Initialization* and *Update*. In the following, we analyze in more detail the overhead introduced by these operations.

5.1. Initialization

Privacy policies $P(A, C)$ are defined by the content owner o in order to allow users in set A to access the contents in set C. To protect the confidentiality of the published contents, each privacy policy needs an initialization phase before being properly enforced. In general, the initialization phase concerns the creation of proper cryptographic data structures, as detailed in the following for each DOSN.

Diaspora. In Diaspora, initialization of a privacy policy does not require any additional costs because storage of data on pods is not encrypted [61]. Consequently, the pod administrator can access all the profile data hosted by the pod and all the data published by users. For this reason, several organizations and users prefer to run their own pod because this provides them more privacy

Table 2
Notations of the different aspects affecting the performance and complexity of a secure DOSN.

Term	Description
$P(A, C)$	a privacy policy authorizing users A to access contents in C
A	the set of authorized users $A = \{a_1, \ldots, a_n\}$
n	the number of authorized users involved in a privacy policy
C	the set of contents $C = \{c_1, \ldots, c_m\}$ protected by a privacy policy
m	the number of contents protected by a privacy policy
GenKeyS	time required to generate a symmetric key
GenKeyAS	time required to generate an asymmetric key pair
GenKeyABE	time required to generate a key with attributes for ABE
EncS	time taken by a symmetric schema for the encryption of the data
EncAS	time taken by an asymmetric schema for the encryption of the data
EncABE	time taken by ABE for the encryption of the data

and control over their data. The communication between pods is always encrypted (using SSL) and the Diaspora protocol uses HTTPS as transfer mechanism between pods. Instead, the communication between the pods and the users can support different levels of confidentiality where data are can be communicated with or without encryption.

Safebook. Safebook ensures the confidentiality of the protected data by leveraging asymmetric and symmetric cryptography. Each registered user is identified by a public–private key pair [32]. Contents C shared for a group of authorized users A are encrypted with a symmetric *data encryption key* (DEK). Furthermore, the owner generates a *key encryption key* (KEK) which is previously distributed among the members of A using their individual public-key. The DEK related to the contents C is encrypted by using the *key encryption key* (KEK) and distributed among all users A that are authorized to decrypt the contents of C [62].

PeerSoN. PeerSoN [21,33] initializes a privacy policy $P(A, C)$ by both exploiting symmetric and asymmetric encryption. Each user in the authorization set A is paired with an individual asymmetric key while contents in C are encrypted with distinct symmetric keys and distributed to the authorized users (or stored on the DHT). In order to protect the confidentiality of contents, each symmetric key of a content in C is securely distributed to the set of authorized users A by encrypting it with their individual asymmetric public keys.

LotusNet. In LotusNet, each user has a pair of RSA keys and an OpenId[3] account. A certification service validates the account and produces a signed certificate containing the user's OpenId and a public key. Each authorized user in A is paired with a grant certificate. Since grants do not hide the published contents in C from the peers that store them (i.e., the peers of the DHT), contents of C are encrypted with a unique symmetric key which is shared with the set of authorized users A by using their asymmetric public key. Indeed, the authors suggest the encryption of the full set of contents in C with a single encryption key. When a replica peer receives a request for a protected content, it verifies the identity of the querying peer by asking for a valid grant. If a valid certificate is provided by the applicant, then the replica peer returns the requested encrypted contents. Finally, the applicant uses their asymmetric private key to obtain the symmetric content key previously shared by the content owner and decrypts the related content with it.

SuperNova. We recall that in Supernova contents are replicated on *storekeepers*, list of users who have agreed to keep a replica. These nodes are not necessarily authorized to access the content. SuperNova [35] uses a cryptographic storage system [63] to enable secure storage on these untrusted nodes. Contents are organized in *filegroups*, i.e., groups of files with the same privacy policies. Filegroups are protected using a symmetric (DES) key, called file-block key, which is exchanged on-demand (via a secure channel). When users want to access a content, they contact the content owner (or other readers), in order to obtain the relevant key. Every file is divided into several blocks where each block is encrypted with a symmetric key. A traditional (k, n)-threshold based secret sharing protocol [64] is exploited to split the contents into n parts where only k of them are required to reconstruct the secret. The authors propose to use this schema for delegating access control and key distribution.

LifeSocial.KOM. LifeSocial.KOM exploits Access Control Lists (ACL) to enable a fine-grained access control. Indeed, compared to Capability Lists, ACL is the most suitable solution in a content centric network. In LifeSocial.KOM [20,23], each user belonging to the authorization set A of the ACL is paired with an individual asymmetric RSA key (1024-bits key length). The public key is used to uniquely identify and authenticate users. A single symmetric AES key (128-bits key length) is created for all the contents published in C and each content $c \in C$ is encrypted, individually, with the symmetric content key. In turn, the symmetric content key is encrypted with the individual public key of each of the authorized users in A, and the resulting list of encrypted keys is attached to the encrypted content. The resulting item, signed by the owner of the content, contains all the information to enforce access control and may be stored on any peers of the DOSN. Authors of [23] analyzed the overhead (in terms of time and storage) introduced by the enforcement of the access control policies, for different number of authorized users (from 1 to 200). The cryptographic mechanisms affect the traffic speed or the storage space by introducing an overhead of about 2 KB on each stored content. Each additional privileged user introduces a data overhead of about 413 bytes and encryption takes 89 ms for 200 privileged users.

Vis-a-Vis. We recall that Vis-a-Vis assumes that users have chosen an external service provider (VIS) which stores and maintains their data available to other users. A basic assumption [37] is that users trust their storage services. Hence, the contents of the users are stored unencrypted on their Virtual Individual Servers (VIS). This basic assumption is based on the observation that the could providers' business model does not allow third parties to exploit the contents produced by their users, like Facebook or Twitter, but it is focused on providing on demand computational resources to users. Users are identified by a self-signed key pair. The private keys of the users are securely stored by their VISs, thus allowing the VISs to act as a proxy for the users. The DOSN requires that users properly configure the privacy policies on their providers. In particular, the VIS of the group founder initially manages the privacy policies of that group. The membership management may be dynamically delegated to other member of the group, during the group lifetime. The IP address of the owner and the owner's public key are distributed out of band.

[3] http://openid.net/.

My3. In order to avoid encryption mechanisms for access control and content storage, users' data are stored unencrypted on the devices of their trusted friends (trust proxy set) [38]. The trust proxy set is directly defined by each user and peers of the trust proxy set must enforce the privacy policy on the contents behalf of the user.

Cachet. In Cachet [27], users' data are protected by EASiER [65]: a cryptographic hybrid structure where privacy policies $P(A, C)$ are enforced by using traditional public–private key and Attribute-Based Encryption (ABE) [60]. Policies are defined by the owner at the time of content creation. Each content of C is encrypted with a randomly chosen symmetric encryption key and the symmetric key used to encrypt the content is encrypted with ABE secret key related to the attributes used in the privacy policy. Users are paired with ABE user keys which specify the values of the attributes characterizing them. Users who do not satisfy the attributes specified by the ABE secret key in the privacy policy cannot decrypt the content. For instance, a user may define the attributes (friend, colleague, neighbor), and generate keys for interesting combination of attributes (e.g., colleague $\wedge$ friend, neighbor $\wedge$ colleague). Then, a user can assign these keys to other users and encrypt the contents with a proper ABE policy. After the contents have been retrieved, they are cached unencrypted on the host of the applicant.

Persona. In Persona, each user initializes a privacy policy $P(A, C)$ by generating a new symmetric content key for each content of C. Each symmetric content key is encrypted either by using a traditional public key or ABE schema. In the former case, the user sends to the members of the group A the symmetric key encrypted with the public key of each member. In addition, the group key may be asymmetric in the case of the group owner wants to allow users which are not members of the group to encrypt messages for the group as well. In the latter case, the symmetric content key of a content is encrypted with an ABE secret key which grants access only to the users having specific attributes' values. The encrypted contents are stored on a specified storage service which make them available to the authorized users. The integrity of the contents is not ensured since Persona assumes that storage services are not interested in tampering with users' data. In addition, Gunnar Kreitz et al. [66] focus on the problem of ABE cryptographic primitives that hide the user's data but reveal access policies. They introduce predicate encryption (PE) [67] in order to hide the user's data without revealing the access policies. A user's profile is defined as a set of multiple objects encrypted for different users and Bloom filter is used to store users who can decrypt the objects.

eXO. The contents shared by users of eXO are stored unencrypted only on the peers of the owners of such contents [40]. The content owner decides whether the added user can access the contents already published. Similarly, the content owner can decide whether still allowing a removed user to access the contents published before his removal. In addition, authors claim that users can decide autonomously to replicate contents on the set of the adjacent peers in the DHT. However, they do not specify if the content on the DHT is stored encrypted or not.

Vegas. In Vegas [41], each user owns an asymmetric key pair for each of his friends. When a user B establishes a friendship relation with another user S, the user B creates a new public–private key for S and he sends the related public key to S. In the same way, user B receives the public key for him created by user S. As a result, a user with n friends has to manage $2n$ public keys and n private keys. Initialization of a privacy policy $P(A, C)$ requires the encryption of each content of C with a new individual symmetric key, which is securely distributed to the users of A by using their public keys.

DiDuSoNet. In DiDuSoNet [42], a privacy policy $P(A, C)$ simply enables access the users' content to all the friends. The contents in C related to a privacy policy $P(A, C)$ are stored unencrypted on the peers of trusted friends (i.e., friends of the content owner). Trusted friends are in charge of regulating access control by providing contents only to authorized users and they are automatically derived from the OSNs by considering tie strength [68,69] of the relationships (in terms of amount of interactions).

Prometheus. Prometheus [70] enforces a privacy policy $P(A, C)$ by exploiting public/private schema, access control lists, and trusted peers. At registration time, the user creates an individual public/private key pair which is used to authenticate the user's account and to protect the cryptographic keys shared by user. In addition, each user specifies the trusted peers that will contribute to manage his contents and creates a unique asymmetric group key for the trusted peers. The asymmetric group key is sent to each trusted peer by encrypting it with the public key of the owner. Each content created by a user is paired to a privacy policy $P(A, C)$ and it is securely sent to the trusted peers by using the public group key. As a result, the trusted peers can decrypt the user's contents. The policy of the contents are encrypted with the public group key and stored on the DHT. In addition, privacy policies are also stored on the trusted peers selected by the content owner. The trusted peers enforce the privacy policies when the corresponding content is requested.

Gemstone. Gemstone [43] exploits both symmetric encryption and ABE to protect the privacy of contents. Initially, users assign attributes to their friends, create new ABE keys based on these attributes, and send the keys to their friends. Given a privacy policy $P(A, C)$, each content of C is encrypted by using a new symmetric key. Finally, each symmetric key is encrypted, by using ABE, with a combination of attributes so that only users who have the required attribute can decrypt it.

Friendica. The users of Friendica [10] are uniquely identified by exploiting OpenId, an open standard and decentralized authentication protocol. Each content published by a user is controlled by four access lists that specify the individuals authorized/unauthorized to access the content, and the groups authorized/unauthorized to access the content. Each user is paired to a user-name and password which are used to log in the system. The contents published by a user are sent to the corresponding Friendica server through a secure channel. When a content c is shared, the Friendica server of the content owner propagates c to the Friendica servers of the recipients. Traffic between Friendica servers can be encrypted depending on the configuration of the involved servers. The contents along with their privacy policies are stored unencrypted on the database of the Friendica servers.

RetroShare. Users of RetroShare are identified by using a public PGP certificate (with 4096 bits RSA key), which provides a web of trust between friends. In order to establish a friendship relation, the involved users must exchange their PGP certificates. Optionally, they can also sign the keys of their friends in order to approve the friends' identities and provide a higher level of trust for friends. After the creation of a new profile, a user can register one or more devices where a RetroShare instance is running by creating a unique SSL certificate for each device. Each SSL certificate is signed with the PGP key, encrypted with the PGP key (along with the SSL passphrase), and stored on the corresponding peer. When the user u logs in RetroShare, u must provide the PGP passphrase in order to decrypt the SSL passphrases by using the private PGP key. The unencrypted SSL passphrase and the SSL certificate are both used to initialize secure communication between friends.

In RetroShare, the content published by a user correspond to shared folders which are located on their local device. All files in

this folder will be hashed (with SHA1) and a special link item is created. The link is used to share the (hash of the) content, to access the collection paired to the content, and to download all files linked to it. Due to the hash of files, the shared folders will not share immediately but after 2 min or up to 1 h, depending on the number of files in the shared folder. The list of shared folders, as well as the friends' lists are stored encrypted on the device of the owner by using the SSL private key of the device. For storage efficiency, the total number of shared files and directories is limited 4,194,303 for a maximum number of 1023 friends. Since contents are stored unencrypted on the users' devices, the administrators of RetroShare recommend to encrypt the home directory of the devices, so that SSL certificates and PGP keys cannot be retrieved by exploiting brute force [71].

Contrail. Users of Contrail are identified by exploiting their user id and their asymmetric key pair. Firstly, the user has to edit his white-list by adding the identifiers of authorized members. In order to ensure delivery of the contents, the filters of the authorized users must be installed on the device of the content producer [72]. A message intended to an individual user is encrypted with a symmetric key that was shared through the public keys of the users at the moment of the creation of the friendship relation.

Instead, the contents shared with multiple authorized recipients are encrypted with a freshly symmetric key and then include this symmetric key is encrypted separately with the public key of each authorized member. In addition, the symmetric keys used to encrypt contents are cached and reused when other contents are sent to the same set of interested users. Authentication of the contents is achieved hash of the content and asymmetric signature.

Soup. Users of Soup [45] are uniquely identified by exploiting 1024-bit public key. Confidentiality of the contents are ensured by securing a content with an ABE secret key which grants access only to the users having specific attributes' values. The encrypted contents are stored on the selected mirror nodes which make them available to the authorized users and cannot access them.

ProofBook. In ProofBook, each user can create a privacy policy that grants access to either a group of users or to an individual friend. Each user's device is uniquely identified by using a public/private key pair which is used for securely exchanging either the symmetric group keys which correspond to different group defined by the user or the individual symmetric key of the friend. In the first case, the symmetric group key is shared with all the authorized members while in the latter case the symmetric individual key is shared only with a specific friend. In both cases, the symmetric keys are exchanged when a user establishes a new relationship and the public key of the authorized user is exploited to protect the communications. Each content created by a user is encapsulated in a container structure which includes the identity of the content owner, a signature of the content, and other useful information related to the content.

DECENT. Contents of DECENT's users are stored on untrusted peers and privacy policy are enforced by exploiting both symmetric (AES) and ABE schema [47]. Each user generates an ABE public, the master secret keys, and signature key pair for the policies. The user generates an ABE secret key which meets the attributes of the privacy policy and distributes the ABE key out of band. The contents C created by the user for their friends are individually encrypted with a random symmetric content key, which governs who can read the content. The write and the append policies are enforced by exploiting both cryptography and specialized DHT functionality. The content owner creates a reference to the content in his profile and he encrypts the symmetric content key for the correct attributes. Indeed, write and append requests are enforced only if sender has successfully produced the corresponding signature based on public/private (RSA) schema. When a user wants to read the content, he finds the reference to the content and decrypts the symmetric content key with his ABE secret key obtained from the content owner. Then the user can retrieve the object from the DHT and decrypts the encrypted fields using the symmetric content key. Integrity and authenticity of the contents is ensured by exploiting DSA signatures.

SocialGate. The contents of the SocialGate [48] users are securely stored on their private gateways or mirror servers by exploiting ABE. As in the case of DECENT, each user generates an ABE public and the master secret keys. The previous keys are exploited by user in order to generate an ABE secret key which meets the attributes of the privacy policy. The authors do not specify how the ABE key are distributed to the authorized users and we assume that they are exchanged out of band. The contents C created by the user for their friends are individually encrypted with a random symmetric content key, which is in turn encrypted with the proper ABE key.

5.2. Updating privacy policies

DOSNs allow their users to update the privacy policies $P(A, C)$ they defined. In particular, at a given time t, a user can change the set A of his contacts allowed to access his contents by adding a new user u or removing an existing one w. In this case, A' is the set of updated members (where, respectively, $A' = A \cup \{u\}$ or $A' = A \backslash \{w\}$), while C is the set of contents that have been published before time t and C' is the set of new contents that will be published after time t. When the set A of a privacy policy is updated, each DOSN adopts its own strategy to manage the permissions on the new contents in C' and on the contents in C. In order to classify such strategies, in the following we define three properties, and we investigate whether each of the considered DOSN ensures them or not. If the privacy policy $P(A, C)$ is updated by adding a new user u, we say that the DOSN satisfies the *Backward Secrecy property* if $P(A, C) \rightarrow P(A, C) + P(A \cup \{u\}, C')$, i.e., the new member u cannot access the contents already belonging to C before the policy update while the old members of A can still access such contents. The new contents that will be published after the policy update (the ones in C') can be accessed by both the new user u and the old members in A. Instead, the *Backward Secrecy property* is not guaranteed when $P(A, C) \rightarrow P(A \cup \{u\}, C) + P(A \cup \{u\}, C')$ where C and C' include the contents published before and after time t, respectively.

Similarly, in case a user $w \in A$ is removed from A, we say that the DOSN ensures the *Forward Secrecy property* only when $P(A, C) \rightarrow P(A, C) + P(A \setminus \{w\}, C')$. In such a case, none of the contents that will be published in C' after the removal of w from A will be disclosed to w because w is not an authorized user any longer. In addition, we say that the *Backward Right Revocation* property was not ensured in the previous definition because the user w can still access the contents in C, i.e., the contents which have been published before his removal. Instead, when $P(A, C) \rightarrow P(A \setminus \{w\}, C) + P(A \setminus \{w\}, C')$ we say that the DOSN also ensures the *Backward Right Revocation property* because the contents that are in C are not accessible to w any longer after his removal from A.

The implementation of the policy update operations could affect the performance of the DOSN system as new cryptographic keys could be generated and some encryption/decryption operations could be performed in order to properly enforce the requirements above. Hence, in the following of this section, we describe how the policy update operations are implemented in the main DOSNs.

Diaspora. In Diaspora, once a user u has published some contents, he cannot change the set of aspects he allowed to access them [61]. Hence, the modification of a privacy policy is not permitted in Diaspora. Indeed, the development team suggest to make a new version of the content and share it to a different aspect. Instead, users can add new members or remove exiting members from the aspects they defined. When a new user is added to an aspect, such user can ask for all the contents that have been published for that aspect. As a result, the backward secrecy property is not guaranteed. When a user is removed from an aspect, he cannot access new contents published for that aspect (i.e., forward secrecy is enforced) because the removed user will be no longer considered when new contents will be published. In addition, Diaspora guarantees the backward right revocation property because the removed user cannot access anymore the contents previously published for that aspect. However, the enforcement of this property on the already existing contents cannot be fully guaranteed because if the removed user is a pod administrator, he can still access old contents of the aspect stored in his local memory.

Safebook. In Safebook [31,62], in order to remove a user from A, the symmetric DEK key of each of the contents in C is refreshed and distributed to the current members of A. However, the contents in C will not be encrypted again with the new DEK key and will still be accessible by the removed user as long as they are not modified. Indeed, the new DEK key will be used to encrypt the new contents, while the existing contents will be encrypted with the new key only in case they are updated by the content owner. As a result, in case of removal of an authorized user from the privacy policy, Safebook ensures the forward secrecy property, but it does not ensure the backward right revocation property. In the case of the addition of a new member to the set of authorized users A, the symmetric DEK keys used to encrypt the contents of C will not be changed and DEK keys are securely distributed to the new user by using his public key and the KEK key. As a result, the addition of a new user to A does not ensure the backward secrecy property, because new members of A are able to access the data previously published in C.

PeerSoN. To add a new user to A, in PeerSoN [21,33], the symmetric keys used to encrypt the contents in C are encrypted with the public key of the new user. In this way, the new authorized users are able to access also the existing contents. Hence, the backward secrecy property is not guaranteed. In the case of user removal from A, the contents already published in C before the removal of user are decrypted and re-encryption with a new symmetric key, which must be no longer accessible to the removed user. Moreover, the key used to encrypt the contents that will be published after the user removal will not be made available to the removed user. As a result, the revocation of the access to a user ensures both the backward right revocation and the forward secrecy properties. A. Datta et al. [64] propose to use threshold-based scheme to address the problem of backup and recovery of the user's private key in a network of untrusted servers. To improve security of the secret sharing protocol they propose a mechanism to select the most trustworthy delegates based on the social relationships between users.

LotusNet. In LotusNet users are able to change the relative grant certificates in order to grant access to new members not in A or deny access to existing members removing them from A [34]. When a new user is added to A a grant certificate is created and distributed to the new user. The contents published before the addition of the user remains encrypted with the same symmetric content keys, which are securely distributed to the joining user by using their asymmetric individual key. A user asking for a content must provide a valid grant certificate in order to download all the encrypted contents published in C, this decrypting the contents with the corresponding symmetric content key. As a result, the backward secrecy property is not ensured by LotusNet because the new member can access all the contents published before the modification of the privacy policy.

In contrast, when a member is removed from A the removed member u is not allowed to download the new data published in C' because he has not a valid certificate. As a result, the forward secrecy property is guaranteed. However, the user u can still access the contents in C published before his removal because u holds the symmetric keys of such contents. As a result, the backward right revocation property is no ensured. The authors propose a solution based on a lazy revocation schema, i.e., the contents already in C when u was removed from A are re-encrypted with the new symmetric key only when an authorized member modifies them. However, this solution is not effective because the majority of contents shared in OSNs are never modified [34].

SuperNova. Users of SuperNova [35] are able to change a privacy policy $P(A, C)$ by adding the identity of a new member to the set of authorizing users A [63]. In such a case, the new member is enabled to request (via a secure channel) the symmetric file-block key of the contents in C and uses it to decrypt each content. Consequently, when the policy is modified for granting the access to new members, Supernova does not ensure the backward secrecy property because the new members can access all the contents published before their join. SuperNova allows the owners of contents to revoke the rights to access the contents already published (i.e., in C) to some users by following a lazy-revocation. In this approach the member is removed from the set of authorized users A but the file-block key used to encrypt the contents in C is changed only when the content is updated. In particular, for each revocation a new version of the filegroup is generated. The new filegroup version contains the updated contents, re-encrypted with a new file-block key. The new file-block key is exchanged on-demand and revoked users can still read unchanged contents in C but they are not able to read both updated contents or new contents published after their removal. Hence, Supernova ensure the forward secrecy property but it does not fully guarantee the backward right revocation property.

Lifesocial.KOM. In Lifesocial.KOM, a privacy policy $P(A, C)$ can be modified by granting access to a new member u [23]. In this case, each content of C is modified, by appending to the list of authorized users the symmetric key of the content encrypted with the individual public key of the new authorized member. By default, the backward secrecy is not ensured because the contents published before u was added to A are accessible to u. In order to revoke access right on contents in C to an authorized user in A, each content of C is modified by removing from the list of authorized users the key related to the removed member. However, it is possible that the removed users have stored the symmetric content key on their local peers to access the contents whenever they want, even if the content owner has denied access to them. For this reason, the affected contents are re-encrypted with a new symmetric key. As a result, the removed user cannot access old contents published before his removal because they are encrypted with a new symmetric content key which is shared only with the authorized members, thus ensuring the backward right revocation property. In addition, each new content published by the user is encrypted with a new symmetric content key which is shared only with the member left in A'.

Vis-a-Vis. Vis-a-Vis [37] assumes that users have chosen an external provider (VIS) for the storage service. Access control is delegated to users, that have protected their contents by properly configuring the privacy policies with their providers.

My3. My3 avoids the enforcement of privacy policies by using encryption mechanisms and exploits trust friends for access control and storage of unencrypted contents [38]. As a result, changes in privacy policies are directly communicated to the set of trusted proxy, which enforce them behalf of the user. Backward secrecy can be ensured to prevent new members to access the contents published before they join the set of authorized users, although no specific mechanisms are provided. Forward secrecy can be ensured in the same way as well. However, when the removed members belong to the set of trusted proxies, they will be still able to see some old contents because a copy of those contents is stored on their devices. As a result, My3 does not ensure the backward right revocation property.

Cachet. Users of Cachet [27] are able to modify their privacy policies $P(A, C)$ in order to grant access to a new user or to revoke access to previously authorized members [65]. Cachet assumes that each user has obtained a fresh individual ABE key which contains the updated values of their attributes. Each content of C is encrypted, individually, with a symmetric key and the symmetric key is encrypted with an ABE content key where access policy is attached to the encrypted item. As a result, the addition of a new member requires the creation of an ABE user key whose attributes satisfy those of the privacy policy. The backward secrecy is not ensured because new member can access all the contents that meet its attributes, even if these contents have been published before the new member was added to A. Users are also able to change the privacy policies $P(A, C)$ of the contents already published in order to deny access to a previously authorized member. In this case, the ABE content key must be refreshed to consider the new access policy which is used to shared future contents with the members left in A'. However, this is not enough to ensure that contents already published in C cannot be accessed by the removed users because they can store locally the symmetric key used to encrypt the contents C. As a result, the backward right revocation property is ensured by refreshing all the symmetric keys used to encrypt the contents of C and by encrypting these with the ABE content key related to the new privacy policy.

Persona. In Persona, a privacy policy $P(A, C)$ defined by a user can accept new authorized members whose attributes' values satisfy those of the ABE access policy. For this purpose, the new member obtains an ABE key which meets the attributes defined by the privacy policy and use it to decrypt the contents of C. The backward secrecy cannot be properly guaranteed because all the contents in C can be accessed by the new member. When a user is removed from A, Persona ensures the backward right revocation property by re-encrypting all the contents already in C with a new individual symmetric key. Moreover, it is necessary to encrypt the new individual symmetric key with an ABE access policy that meets the attributes defined by the privacy policy. By using this approach, the removed member will no longer be able to obtain the symmetric key of the contents in C, thus enforcing backward right revocation property. In addition, the removed user cannot access future contents because their symmetric individual keys are encrypted with an ABE access policy that meets the attributes defined by the new privacy policy, thus enforcing the forward secrecy.

eXO. Users of eXO can store unencrypted contents on their local peers and changes in privacy policies are directly enforced by the content owner [40]. Since contents are stored on the local device of the content owner, users are able to prevent both new members from accessing old contents (backward secrecy property) and the removed members to accessing the new contents (forward secrecy). Eventually, the backward right revocation property with respect the removal of a user can be directly guaranteed by the content owner, which may decide whether to share or not contents already published with the removed user.

Vegas. When the user changes a privacy policy $P(A, C)$ by adding a new member to the set of authorized user A, he has to notify all the contents in C to the new member. For this reason, the symmetric key of each content is encrypted with the individual public key of the new user and it is sent to him/her. By default, backward secrecy is not ensured but, eventually, it can be guaranteed by disclosing only the symmetric keys of the new contents in C published after the join of the new user [41]. Instead, key revocation introduces more overhead since it requires the re-encryption of each content with a new individual symmetric key and the distribution of such keys to the new members (by using their asymmetric public keys). Therefore, backward right revocation property is ensured because the removed member can no longer access old contents in C.

DiDuSoNet. In DiDuSoNet [42] changes in privacy policies are directly enforced by the content owner and do not introduce any overhead because contents are stored unencrypted on the peers of trusted friends. By default, the backward secrecy property with respect to user's addition is not guaranteed because the new member can access all the contents published after their join. Instead, DiDuSoNet ensures the forward secrecy property when a privacy policy is updated by denying access to a member because the removed user cannot anymore access old contents. In addition, the backward right revocation property is not ensured because it requires the reallocation of the contents on the new users' peers who are authorized to access them.

Prometheus. The user of the Prometheus [70] are able to modify a privacy policy $P(A, C)$ by adding a new member to the set of authorized users. For this reason, the content's owner executes a three-way handshake procedure that allows the secure sharing of the public/private group key with the new member. The public/private group key is used to encrypt content of C. As a result, the new members will be able to decrypt all the contents of the privacy policy by using the public/private key exchanged in the previous step. The backward secrecy is not guaranteed because the new member is able to access all the contents published in C. When a user decides to remove a member from the trusted group of a privacy policy, he submits an unsubscribe multicast request to all members of the privacy policy (except for the removed user). The affected users' peers generate a new public/private group key for the privacy policy which is distributed to all the authorized users. The backward right revocation property is not guaranteed because old contents remain encrypted with the old group key and they can still be accessed by the removed user.

Gemstone. In Gemstone [43], new members having ABE key whose attributes satisfy those of a privacy policy $P(A, C)$ are authorized to access the contents of C. The backward secrecy cannot be property guaranteed because all the contents in C remain visible to the new member. Instead, deny access to a member of A requires the re-encryption of each content of C with a new individual symmetric key in order to avoid disclosure of old contents. In addition, each symmetric key is encrypted with ABE by using an access policy which meets the attributes defined by the privacy policy. In this case, the removed member will no longer be able to obtain the symmetric key of both old and new contents, thus preventing both forward secrecy and backward right revocation property.

Friendica. In Friendica, users are able to change the composition of their groups by removing members or adding new ones to the access control list of the groups. When a new member is added to the group, he will be able to access future contents and old contents published in the group. As a result, the backward secrecy property is not guaranteed in the case of user addition. Instead, a user removed from a group cannot access future contents that will be published in the group (forward secrecy), but he can still access old contents already published in the group because users

authorized to access the content have permanent permissions. As a result, the backward right revocation property is not guaranteed because, once a user has created a content and shared it with a group, the content has been delivered to the Friendica servers of the recipients. For this reason, the content owner cannot anymore change the privacy policy assigned to the content by restricting access to some users in the group. In such a case, the Friendica suggests to delete the content by sending a delete notification to everybody who received the content [73].

RetroShare. Users of RetroShare are able to edit permissions on contents shared with their contacts. Privacy policies on shared contents are directly enforced by the content owner, when the content is requested by a member. As a result, the peer of the content owner ensures that new member of the group can access the contents that will be published for that group. The backward secrecy property is not provided because the new member can access also the contents already published in the shared directory. Similarly, revoking access right to a user is directly enforced by the peer of the content owner, which denies the access to the removed user when he requests the contents (i.e., enforcing forward secrecy). However, deny access to a member does not ensure the backward right revocation property because it is possible that the users removed from a group have stored a copy of all the contents in their local storage.

Contrail. In Contrail, users are able to change the set of authorized member by removing members or adding new ones to the white-list. In addition, the corresponding filters must be installed (or removed) from the user's device. When a new member is added, he will be able to access future contents but he cannot access old contents published in the user's profile. As a result, the backward secrecy property is guaranteed in the case of user addition. Instead, a user removed from the white-list cannot access future contents that will be published by user in his profile (forward secrecy), but he can still access old contents already published in by the profile owner because they are permanently moved from the cloud to the users' devices. As a result, the backward right revocation property is not guaranteed. However, it is possible to set expiry times on contents in order to delete the contents from the cloud's storage if the recipients of the contents do not connect to the system before the end of the expire time.

Soup. Authors of Soup [45] do not specify how the privacy policy $P(A, C)$ defined by a user can be updated. For this reason, we assume that they are enforced by using a strategy that is similar to other DOSNs based on ABE schema (such as Persona, Gemstone, or Cachet). In particular, a privacy policy $P(A, C)$ can accept new authorized members whose attributes' values satisfy those of the ABE access policy. For this purpose, the new member obtains an ABE key which meets the attributes defined by the privacy policy and use it to decrypt the contents of C. The backward secrecy cannot be properly guaranteed because all the contents in C can be accessed by the new member. Instead, deny access to a member of A requires the re-encryption of each content of C with a new individual symmetric key in order to avoid disclosure of old contents in order to ensure the backward right revocation property. Moreover, it is necessary to encrypt the new individual symmetric key with an ABE access policy that meets the attributes defined by the privacy policy. By using this approach, the removed member will no longer be able to obtain the symmetric key of the contents in C.

ProofBook. To add a new user to A, in ProofBook [46], the symmetric group key used to encrypt the contents in C are encrypted with the public key of the new user. The new authorized users are able to access also the existing contents, by requesting them to the other authorized members or to the content owner. Hence, the backward secrecy property is not guaranteed. In the case of user removal from the group A of authorized members, a new symmetric group key must be created and shared only with the members of the group. In addition, the new symmetric group key must be no longer accessible to the removed user. The contents already published in C before the removal of user cannot be re-encrypted with a new symmetric group key because they have already been distributed to different authorized peers. As a result, the revocation of the access to a user does not ensure the backward right revocation. However, the future contents and the update of old contents will no longer accessible to the removed users, hence enforcing the forward secrecy properties.

DECENT. When the user changes a privacy policy $P(A, C)$ by adding a new member to the set of authorized user A, he has to generate a ABE decryption key for the new user that meets the attributes define by the privacy policy. ABE keys are exchanged out of band and only the keys that satisfy the attributes of the encrypted data can obtain the corresponding contents. As a result, the backward secrecy is not ensured [47]. Instead, the removal of a user form A requires to revoke one or more attributes from the friend. The schema adopted by DECENT ensures that the revoked contact cannot access any contents that requires the attributes, hence enforcing forward secrecy property. In addition, the contents already published in C cannot be accessed by the removed user, i.e. enforcing backward right revocation property. In order to achieve these goals, DECENT exploits the EASIER schema [65] where introduce a minimally trusted proxy is introduced to block access to previously published contents, without re-encrypting them. A user who wants to decrypt a content takes a part of the encrypted content to the proxy. Each proxy has a secret proxy key with revocation information which is used to transform the part of the content and this transformation is required by users to successfully decrypt the content. When the content owner revokes access to a user, he has to update also the secret proxy key so that the revoked user cannot exploit the transformation to successfully decrypt the content.

SocialGate. Users of SocialGate [48] can change a privacy policy $P(A, C)$ by adding a new member to the set of authorized user A. For this purpose, the content owner has to generate a ABE decryption key for the new user that meets the attributes define by the privacy policy. ABE keys are exchanged out of band and the ABE keys that satisfy the attributes can obtain all the contents intended for these attributes. As a result, the backward secrecy is not ensured. Instead, the removal of a user form A requires to revoke one or more attributes from the friend. The removed contact cannot access any contents that requires the attributes, hence enforcing forward secrecy property. However, the contents already published in C can be accessed by the removed user, because they are encrypted with the old attributes. As a result, SocialGate does not enforce backward right revocation property. In order to achieve this goal, a user has to decrypts the contents already published on the gateway and re-encrypts these contents with the new privacy policy.

6. Evaluation

The previous sections surveyed some crucial aspects of current DOSNs, and this section presents a comparison among them with respect to those aspects. In particular, we analyze the privacy models provided by DOSNs and evaluate the overhead introduced for privacy policy management.

6.1. Evaluation of the privacy models

To help the reader in understanding the different types of privacy models provided by current DOSNs, previously described in Section 4, we propose to classify them by using the taxonomy shown in Fig. 2. In particular, we identified 4 different privacy models:

Relationship-based: where the relationships (such as friendship) established by users, as well as the features of these relationships, are directly exploited by the DOSN users in order to define their privacy policies.

Group-based: where users are able to organize their contacts in a set of groups, and they define their privacy policies by granting the right to access their contents to these groups.

Profile-based: where each user exploits the profile information of the other users to define their privacy policies.

Content-based: where users organize their contents in distinct groups (or types) and they exploit these groups (or types) to define privacy policies that permit access only to the specified set of contents.

As shown in Table 3, all the considered DOSNs except Diaspora, allow their users to define relationship-based privacy policies. Most of DOSNs, such as Safebook, Cachet, SocialGate, DECENT, Persona, Soup, eXO, Vegas, DiDuSoNet, Prometheus, Gemstone, allow users to organize their contacts in homogeneous groups by specifying the type of relationship (such as family, acquaintances, close friend, colleague, etc.). Then, users can state privacy policies which exploit the type of relationships. In particular, Safebook allows users to assign labels to each relationship in order to define *badges*, i.e., sets of contacts having the same labels.

Besides relationships type, some DOSNs enable users to provide attributes for their relationship. Such attributes are features which can be either automatically derived from the DOSN knowledge or explicitly provided by a user for each of their contacts. For example, the depth of a relationship (such as friend, friend of friend, etc.) is used by Safebook, RetroShare, Soup, and Cachet as attribute of privacy policies.

The identity of a user involved in a friendship relationship (friend's identity) is another attribute of the relationships which can be easily obtained from the DOSN knowledge and it is used by PeerSoN, LotusNet, SuperNova, LifeSocial.KOM, Vis-a-Vis, Cachet, DECENT, SocialGate, Soup, Friendica, ProofBook and Vegas to define privacy policies.

In addition, Safebook and Prometheus give their members the ability to specify how much they trust their friends or if they know personally each person they have a relationship with on the OSN, while DiDuSoNet leverages trust model proposed in the literature to derive and compute the trust value of a user by using the information about friendships and interactions between OSN's members. The nature of this model takes advantage of the important sociological concept of Dunbar Circles [74]: the idea is that friends in an ego network can be described by different levels of intimacy and closeness to the ego. To reflect different levels of importance of these relationships, friendships are associated to a tie strength, a numerical value describing their force. The knowledge of tie strength can be exploited to define privacy policies which exploit this confidence level.

Group-based privacy policies allow users to organize their contacts into distinct groups, namely *groups* in LifeSocial.KOM, Vis-a-Vis, and Persona, *aspects* in Diaspora, *circles or groups* in Safebook, or *filegroup* in PeerSoN. These groups differ from those resulting by the types of the relationships because they contain contacts with different types of relationships. As a result, group resulting from the relationship-based privacy policies are homogeneous in terms of types of relationships while group-based privacy policies are meant for heterogeneous groups.

In most of the cases, the user who created a group can decide whether to make it visible only to himself, to the group members, or to any user of the DOSN. In addition, membership information about a group can be made accessible either to the group owner, to the group members, to any user. Users of Diaspora, Safebook, PeerSoN, Vis-a-Vis, Friendica, RetroShare, ProofBook, and Persona are able to create private groups (named also circles) which are intended to be used only by the user who defined them (i.e., the group owner). As a result, only the group owner is aware of both the existence of the group and of which users belong to it. In Diaspora, public or private groups (named also aspects) are visible only to the group owner while the members can only see the group's name. In Safebook, circles are private groups visible only to the group owner and a member of the group is not aware of the other group's members. In addition, users of Diaspora, LifeSocial.KOM, and Vis-a-Vis are able to create public groups of users focused on specific topics (such as Music, Movie, or Photography). In Diaspora, public groups are visible only to the group owner while the identities of the group members are visible to each other. Instead, in LifeSocial.KOM and Vis-a-Vis public groups information are visible to any users of the DOSN, who may decide to explore and join them. In contrast, public groups created by the users of Safebook are visible to group members, as well as the identities of the members who belong to the group. A similar capability provided by RetroShare is the concept of circle [44], i.e., groups of anonymous identities which can be (i) visible to any friends (Public) (ii) visible only to members (Private), (iii) visible to another circle (Restricted), or (iv) visible only to invited members. Instead, contents published by the users of Contrail can be accessed by all their friends (i.e., the users on the white-list) and a user cannot create other groups. The consumers of the contents can install filters on the device of the group owner in order to indicate his interest in specific contents, based on tags, keywords, or GPS coordinates of the producer.

Content-based privacy policies allow users to attach attributes to the contents in order to exploit them during the definition of privacy policies. For instance, Safebook, LotusNet, and Prometheus model the type of content (such as Post, Comment, Like, or Photo) as attribute, and the privacy policies define access permissions based on content type. The type of content can be automatically derived from the content itself, or manually defined by the user who specifies the content type by means of labels. Other sources of information used to support the design of privacy policies are those related to the user profiles. Typically, OSNs enable their users to define their own profile: a digital representation of the users containing their personal information, interests, school, partner information, political preference, jobs, etc. Profile-based privacy policies allow users to exploit profile information to decide who can access their contents, on the basis of different aspects of the profile. In particular, members of Prometheus, Soup, and Gemstone can use attributes that model features originating from user-entered profile information (such as location and interest). These informations are typically contained in the public part of the users' profiles and they can be exploited by any user of the DOSN in order to limit access to public contents to users having specific interests or location. Finally, Friendica allows users to restrict access to contents based on the DNS location of the applicant.

6.2. Evaluation of the privacy policy management

Since the majority of the current DOSNs enforce the privacy policies defined by their users through cryptography, in the following section we evaluate the overhead introduced by the initialization and modification of a privacy policy $P(A, C)$ in terms of the number of cryptographic keys created (#Key), and the number of encryption operations required (#Enc) when a policy $P(A, C)$ is created and when it is modified by adding or removing members to A. For instance, Prometheus relies on asymmetric encryption while the others combine both asymmetric and symmetric cryptography. There are also a small collection of DOSNs (such as Cachet, DECENT, Persona, and Gemstone) that propose to improve the capabilities of existing approaches by integrating Attribute-based Encryption (ABE) [60]. In Section 6.2.2, instead, we describe some approaches that are not based on cryptography.

Table 3
Classification of the privacy models provided by current DOSNs according to the taxonomy defined by Fig. 2.

DOSN	Group-based			Relationship-based	Profile-based	Content-based
	group name	group visibility	member list			
Diaspora	aspect	group owner group owner	group owner group member			
Safebook	circles public group	group owner group member	group owner group member	type, depth, trust		content type
PeerSoN	filegroup	group owner	group owner	friend's identity		
LotusNet				friend's identity		content type
SuperNova				friend's identity		
LifeSocial.kom	public group	any user	any user	friend's identity		
Vis-a-Vis	private group public group	group member any user	group member any user	friend's identity		
Cachet				friend's identity, type, depth		
Persona	private group	group owner	group owner	type		
eXO				type		
Vegas				friend's identity, type		
DiDuSoNet				type, trust		
Prometheus				type, trust	location	content type
Gemstone				type	location, interest	
Friendica	private group public group public circle	group owner group owner any friend	group owner group owner anonymous	friend's identity	DNS location	
RetroShare	private circle restricted invited only	circle member circle member circle member	anonymous anonymous anonymous	depth		
Contrail	white-list	group owner	group owner		location	tags, keywords
Soup	private group	group owner	group owner	friend's identity, type	location	
ProofBook	private group	group owner	group member	friend's identity		
DECENT				friend's identity, type		
SocialGate				friend's identity, type		

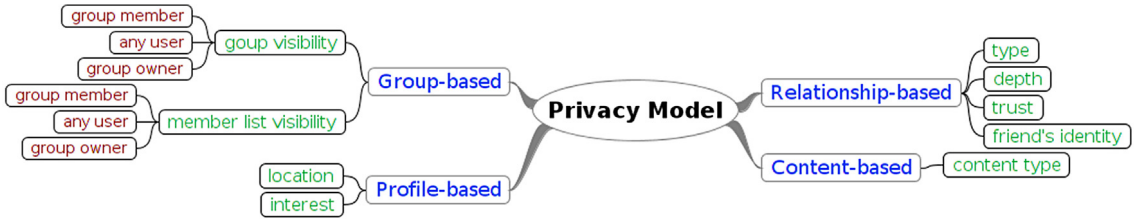

Fig. 2. A taxonomy for the classification of privacy models.

6.2.1. Cryptography-based DOSNs

Initialization. Table 4 shows the costs for the initialization of privacy policies in the DOSNs we examined. In particular, we measured the number of cryptographic keys created in order to protect a content. Please notice that in Table 4 we do not take into account the creation of the personal asymmetric or ABE keys paired to the users because these keys are created only once when the users join the DOSN and they are exchanged with the other users when the relationships are established. In general, it is well known that cryptographic schemes introduce costly operations for the generation of cryptographic keys, for encrypting plain texts, and for decoding cryptograms. Since asymmetric (and ABE) operations are significantly more costly than symmetric ones, for each DOSN, we counted separately the number keys created by using the symmetric schema (*GenKeyS*), the asymmetric schema (*GenKeyAS*), and the attribute-based schema (*GenKeyABE*), as well as we distinguish between the number of encryption operation performed by using the symmetric schema (*EncS*), the asymmetric schema (*EncAS*), and the attribute-based schema (*EncABE*).

Every time a member of the DOSN creates a privacy policy $P(A, C)$ which grants to the n users of A the access to the m contents in C, a set of new keys needs to be generated for protecting the m contents. In particular, the data representing the contents are typically encrypted by using the symmetric schema. In order to ensure fine-grained and efficient access control, most of the current DOSNs (such as PeerSoN, LifeSocial.KOM, Cachet, DECENT, SocialGate, Persona, Vegas, and Gemstone) create a new symmetric key for each content to be protected (for a total of m keys). Then, each of the m contents is encrypted with the corresponding symmetric key (for a total of m symmetric encryption operations). In contrast, Safebook, LotusNet, Contrail, ProofBook, and SuperNova create only one symmetric key, which is used to encrypt all the contents in C. In particular, Safebook requires the creation of two symmetric keys because it exploits a unique symmetric content key to encrypt contents and it is securely distributed to authorized members by using a different symmetric key (key encryption key). A similar approach is exploited also by Prometheus, which creates a new asymmetric key for the group and all the contents are protected by using this key.

The symmetric/asymmetric key(s) used to encrypt the contents of C must be securely distributed to the n authorized users of A. For this purpose, SuperNova exchanges it/them when requested by the authorized users, via a secure channel, while in DECENT these keys out of band. However, the most part of current DOSNs exploit asymmetric encryption for securely distributing the key(s) generate in the previous step. As previously recalled, the public–private key pair of each user is generated only once when the user registers to the DOSNs. As for instance, Prometheus encrypts the contents with the group public asymmetric key and exploits the individual public asymmetric keys of users to securely distribute the private group key to authorized contacts. In contrast, Vegas, creates an individual public–private key pair for each friendship

Table 4
Evaluation of the overhead for privacy policy definition.

DOSN	Initialization	
	#Key	#Enc
Safebook [31]	$2 \cdot GenKeyS$	$m \cdot (2 \cdot EncS) + n \cdot EncAS$
PeerSoN [21]	$m \cdot GenKeyS$	$m \cdot (EncS + n \cdot EncAS)$
LotusNet [34]	$GenKeyS$	$m \cdot (EncS + n \cdot EncAS)$
SuperNova [35]	$GenKeyS$	$m \cdot EncS$
LifeSocial.KOM [20]	$m \cdot GenKeyS$	$m \cdot (EncS + n \cdot EncAS)$
Cachet [27]	$m \cdot GenKeyS$	$m \cdot (EncS + EncABE)$
Persona (ABE) [39]	$m \cdot GenKeyS$	$m \cdot (EncS + EncABE)$
Vegas [41]	$m \cdot GenKeyS$	$m \cdot (EncS + n \cdot EncAS)$
Prometheus [70]	GenKeyAS	$m \cdot EncAS + n \cdot EncAS$
Gemstone [43]	$m \cdot GenKeyS$	$m \cdot (EncS + EncABE)$
Contrail [75]	$GenKeyS$	$m \cdot (EncS + EncAS)$
Soup [45]	$m \cdot GenKeyS$	$m \cdot (EncS + EncABE)$
ProofBook [46]	$GenKeyS$	$m \cdot EncS + n \cdot EncAS$
DECENT [47]	$m \cdot GenKeyS$	$m \cdot (EncS + EncABE)$
SocialGate [48]	$m \cdot GenKeyS$	$m \cdot (EncS + EncABE)$

Table 5
Evaluation of the current approach for user addition.

DOSN	Grant access to a new user		
	#Key	#Enc	BW
Safebook [31]	0	$EncAS + EncS$	✗
PeerSoN [21]	0	$m \cdot EncAS$	✗
LotusNet [34]	0	0	✗
SuperNova [35]	0	0	✗
LifeSocial.KOM [20]	0	$m \cdot EncAS$	✗
Cachet [27]	$GenKeyABE$	0	✗
Persona (ABE) [39]	$GenKeyABE$	0	✗
Vegas [41]	0	$m \cdot EncAS$	✗
Prometheus [70]	0	EncAS	✗
Gemstone [43]	$GenKeyABE$	0	✗
Contrail [75]	0	0	✓
Soup [45]	$GenKeyABE$	0	✗
ProofBook [46]	0	$EncAS$	✗
DECENT [47]	$GenKeyABE$	0	✗
SocialGate [48]	$GenKeyABE$	0	✗

relations and each user u has to manage a total of $2 \cdot f$ public–private key and f private keys, where f is the number of friends of u. In other DOSNs (such as Safebook, PeerSoN, LotusNet, LifeSocial.KOM, ProofBook, and Contrail), each user is linked to a public–private key pair, where the public part of the key is used to uniquely identify the user and it is made available to all their contacts, while the private part is kept secret by the user. If the contents of C are protected by a unique symmetric key, this key is encrypted with the public key of each the n authorized users; alternately, the m symmetric keys used to encrypt the contents of C, are individually encrypted with the public key of each the n authorized users. The resulting list of encrypted keys can be attached to each encrypted content or directly distributed to the authorized users. In any case, n asymmetric encryption operations ($n \cdot EncAS$) are necessary for each symmetric key used to encrypt the contents of C. In this way, the authorized users are able to decrypt the symmetric key(s) used to encrypt the contents of C with their private keys, and they can use such symmetric key(s) for accessing the contents.

Recently, Cachet, DECENT, SocialGate, Persona, Soup, and Gemstone propose to leverage the ABE schema to securely distribute the symmetric keys used to encrypt the contents of C to the n authorized users. To use ABE, each user generates an ABE public key and an ABE master secret key. For each friend, the user can then generate an ABE secret key which is associated with a set of attributes. Attributes define a logical expression that users must satisfy in order to decrypt the data. ABE ensures that only users with the correct attributes will be able to decrypt the data. As a result, each symmetric key is encrypted only once for all the n users, with the proper logical expression over attributes. Although ABE can affect the performance of old devices because operations have proved to be about 100–1000 times slower than those of the RSA [39], it can be exploited by devices of the latest generation. Finally, it is worth noting how the number of encryption operations required by users to publish m contents is the same of the number of encryption operations (i.e., #Enc) needed to initialize the privacy policy in Table 4. Indeed, the content publisher encrypts the m contents with the appropriate symmetric/asymmetric keys and shares them to the authorized users.

Overhead for updating privacy policy. Updating privacy policies is an operation that allows users to redefine their privacy preferences in order to *grant access* to some contents to new users or to *deny access* to previously authorized users. Since we assumed that content confidentiality is enforced through cryptography, when a privacy policy is changed, new cryptographic keys must be generated and some encryption/decryption operations must be performed in order to properly enforce such changes.

For each DOSN, Table 5 shows the number of generated keys (#Key), and the number of encryption/decryption operations (#Enc) required to update the policy $P(A, C)$ to grant the access to a new user. First of all, we notice that the most part of the current DOSNs do not guarantee the backward secrecy property (last column of Table 5, BW). Hence, when the users of such DOSNs change their privacy policies $P(A, C)$ to grant the access to a new user u, besides allowing u to access the future contents that will be added to C, they also enable u to access the contents already published in C. For this reason, these DOSNs (such as Safebook, PeerSoN, LifeSocial.KOM, Prometheus, ProofBook, and Vegas), in order to grant the access to a new user u, share the keys used to protect the m contents of C with u, by encrypting each of them with the individual asymmetric key of u. In the case of ProofBook In particular, Safebook encrypts the symmetric Key Encryption Key (KEK) by using the individual public-key of the new member and, in turn, the KEK is exploited to securely communicate the individual symmetric key used to protect the contents. Instead, in Prometheus, the asymmetric key pair used to encrypt the content is sent to the new user u through a secure channel (TCP-like three-way handshake procedure). Hence, it is not required to further encrypt this key pair. However, the establishment of the secure channel requires an additional cost which is not reported in Table 5. In contrast to these approaches, LotusNet and SuperNova do not incur any cost when the policy is changed, because users have to request the symmetric key of a content when they want to access it, by providing a valid grant certificate.

When ABE schema is used to protect contents (such as in the cases of Cachet, Persona, DECENT, SocialGate, and Gemstone), the cost of granting access to a new user is equal to the cost of creation of the ABE key with the proper attributes' values.

Unlike the most part of current DOSNs, Contrail is the only one that guarantees the backward secrecy property because all contents already published in the DOSNs are removed from the cloud-based relay-servers when downloaded by authorized users. It is interesting to note that it does not really make sense to ensure forward secrecy property for the user addition operation because the new member will be authorized to access the contents that will be published in the group.

Table 6 shows the costs for changing the privacy policy $P(A, C)$ in order to revoke the access right to a user $u \in A$. As for the join operation, Table 6 shows the number of generated keys (#Key), and the number of encryption/decryption operations (#Enc) required to update the policy $P(A, C)$ to deny the access to an authorized member. In addition, we assessed whether the backward right revocation property (BRP) is guaranteed. The aim of the removal of a member from the set of authorized users is exactly to guarantee

Table 6
Evaluation of the current approach for the case of deny access.

DOSN	Deny access to a previously authorized user		
	#Key	#Enc	BRP
Safebook [31]	*GenKeyS*	$n \cdot EncS$	✗
PeerSoN [21]	$m \cdot GenKeyS$	$m \cdot (EncS + n \cdot EncAS)$	✓
LotusNet [34]	0	0	✗
SuperNova [35]	*GenKeyS*	0	✗
LifeSocial.KOM [20]	$m \cdot GenKeyS$	$m \cdot (EncS + n \cdot EncAS)$	✓
Cachet [27]	$m \cdot GenKeyS$	$m \cdot (EncS + EncABE)$	✓
Persona (ABE) [39]	$m \cdot GenKeyS$	$m \cdot (EncS + EncABE)$	✓
Vegas [41]	$m \cdot GenKeyS$	$m \cdot (EndS + n \cdot EncAS)$	✓
Prometheus [70]	*GenKeyAS*	$n \cdot EncAS$	✗
Gemstone [43]	$m \cdot GenKeyS$	$m \cdot (EncS + EncABE)$	✓
Contrail [75]	0	0	✗
Soup [45]	$m \cdot GenKeyS$	$m \cdot (EncS + EncABE)$	✓
ProofBook [46]	*GenKeyS*	$n \cdot EncAS$	✗
DECENT [47]	$GenKeyABE + GenKeyS$	0	✓
SocialGate [48]	*GenKeyABE*	0	✗

that none of the future contents published in C will be disclosed to u. As a result, the forward secrecy property is always guaranteed in case of user the removal because users do not want to share new contents published in the group with the removed members. Indeed, except for LotusNet and SuperNova, the forward secrecy is ensured by generating the new key(s) for encrypting future contents and by distributing such a key only to the updated set of authorized group members. Instead, LotusNet and SuperNova, provides the updated key on-demand to the users that requests the contents.

Some DOSNs ensure the backward right revocation property, i.e., the contents previously published in C are no more accessible to the removed user u. Instead, other DOSNs ensure that previously published contents of C will still be accessible to u.

The majority of existing DOSNs (such as PeerSoN, LifeSocial.KOM, Cachet, DECENT, Persona, Vegas, Gemstone, Soup) that rely on both symmetric and asymmetric (or ABE) schema ensure backward right revocation property in case of removal of a user. For this reason, whenever a user is removed from the set of authorized users A of a privacy policy $P(A, C)$, the symmetric keys of the contents in C must be changed, and the new keys must be redistributed to all the current members of A (obviously, except for the removed member u) by using their individual asymmetric key or ABE key. In this way, disclosure of either new or old contents to the removed user is avoided. A different solution is employed by DECENT, where the update of the symmetric key used by the proxy ensures that the revoked user cannot decrypt the old contents. However, the proxy server must be contacted whenever the users request the contents.

In addition, the previous solution is affected by a serious drawback: it does not scale well for contents shared with large groups of users due to the overhead introduced by encryption mechanisms in terms of number of keys that have to be exchanged, associated encryption/decryption operations, and size of the messages sent [22]. Indeed, the number encryption operations to be executed to remove a user from a group is linear on the number of users belonging to that group. Authors of [22,23,30] showed that some current DOSNs have a cost per user removal from a group proportional to the size of the group.

Two different approaches have been adopted by current DOSNs in order to mitigate with these problems. The first approach consists of not guaranteeing the backward right revocation property, such as Safebook, LotusNet, SuperNova, Contrail, ProofBook, SocialGate, and Prometheus, which allows the removed users to access old contents of the group. Hence, previously published contents remain encrypted with the same cryptographic keys which are known by the removed users. Indeed, DOSNs that do not ensure the backward secrecy property during user's removal have only to delete the user identity from the set of authorized users, and he will no longer be considered as authorized user when new contents will be published, but already existing contents remains encrypted with the same keys or, as in the case of SuperNova, re-encrypted only when the affected contents are updated. A similar approach is used also by Safebook, that allows the removed users to access the same copy of the old contents as long as the content owner does not update them. For what concerns LotusNet, we remark that the cost for user removal is zero because they do not have a real user removal procedure, but they simply assign short validity periods to grant certificates and they do not renew the grant certificates to users who have been removed from the authorized user set. Obviously, in the LotusNet approach, the choice of a proper validity period for the grant certificates is critical for guarantee the privacy of the contents. As a matter of fact, authors suggest the usage of certificates with short duration. Finally, in the case of Contrail, the backward right revocation property is not guaranteed because the contents already published have been removed from the cloud-based relay servers and transferred on the peers of the user removed from the group.

The second approach to mitigate the efficiency problem related to the removal of a user [76,28,77,78] exploits the strength hierarchical data structures for reducing the overhead of the update of the privacy policy. As for instance, authors of [76,28,77] of the Logical Key Hierarchy model (LKH) [79] for managing the update of the key of a group. The LKH model leverages the hierarchical properties of the tree data structure to reduce the number of encryption/decryption operations needed when a member is removed from a group. In particular, the authors of [28] propose an approach where removing a user from a privacy policy $P(A, C)$ requires $O(d \cdot \log_d(n))$ encryption operations where d is the maximum number of children of the nodes and the number of encryption operations depends on the height on the tree. Instead, the join of a user requires only $O(2 \cdot \log_d(n))$ encryption operation. Indeed, the authors of [76] propose a decentralized group key management algorithm which combines both the LKH and the tree-based group Diffie–Hellman (TGDH) where authorized members are managed by different LKH trees which are combined by using TGDH scheme. The join or remove of a user require the update the corresponding LKH tree while TGDH tree is used for inter-group communication.

Finally, the authors of [78] propose to define hierarchy of groups where some subgroups have more or fewer privileges than others. Each group is paired to a symmetric key and some private information. The symmetric key is used to encrypt data while the private information are used to derive the keys assigned to subgroups in the hierarchy. However, the assignment of the private information, as well as of the symmetric key, is performed by a central trusted authority.

Table 7
Alternative solutions to cryptography-based DOSNs.

DOSN	Privacy Policy Management		
	User device	Peers Selection	Scope
Diaspora [9]	not required	trusted by users	internal (pods)
Vis-a-Vis [37]	not required	trusted by users	external (virtual Identification server)
My3 [59]	required	trusted by users	internal (trusted friends' peers)
eXO [40]	required	localhost	internal (localhost and DHT peers)
DiDuSoNet [42]	required	tie strength	internal (derived from trust model)
Friendica [10]	not required	trusted by users	internal (Friendica servers)
RetroShare [11]	required	localhost	internal

6.2.2. Alternative approaches

Besides the ones previously described, some DOSNs such as Diaspora, Vis-a-Vis, My3, eXO, DiDuSoNet, Friendica, and RetroShare avoid the use of cryptography by storing the contents unencrypted on some trusted replica peers. In particular, the replica peers can be: (i) the peer of the content owner, (ii) the peers of the other users explicitly selected by the content owner, (iii) the peers of the friends with higher strength of the relationship, (iv) the peers of the users that are authorized to access the content, based on the privacy policy defined by the content owner.

Consequently, these DOSNs do not have to perform any cryptographic operations to initialize or to update the privacy policies, since the enforcement of such policies is directly performed by the contents owner (or by the trusted peers that behave as proxy on behalf of the content owner) when the other users request to access the contents. Table 7 shows the characteristics of these DOSNs. In particular, we investigated if such DOSNs require users to provide the resources of their devices (column labeled User device), the trust model exploited by the DOSNs in order to select other trusted replicas when necessary (column labeled Peers Selection), and whether the peers selected as replica are managed by users who are registered to the DOSN, i.e., internal, or they are external, i.e., provided by third parties (column labeled Scope). In addition, we summarized in Table 8 whether such DOSNs ensure or not the Backward Secrecy (BW) and the Backward Right Revocation property (BRP). For instance, the DOSNs that store contents C only on the peers of the owner of such contents (such as eXO, and RetroShare) do not need to encrypt them, because the contents are stored on a trusted device of the owner, who is obviously authorized to access them. As a result, enforcement of privacy policy is directly performed by the contents owner when the access to the content is requested. This operational mode also applies to Diaspora and Friendica in the case of users who have decided to run their pods or Friendica servers. In addition, some DOSNs assume that the device of a single user u could not enough to ensure the required availability of the contents published by u. For this reason, My3 and Friendica enable users to select several trusted servers where to store their contents. In the case of DiDuSoNet, the selection of trusted replica peers is dynamically performed by the system by exploiting tie strength between users.

As for the DOSNs based on cryptography, the update of a privacy policy by granting access to a new user does not provide the Backward Secrecy property because the new member is able to access the contents published before his join.

In the case of a policy $P(A, C)$ is updated by removing a user from A, the Backward Right Revocation property cannot always be fully guaranteed because it is possible that the removed user u has stored the contents already published on their local peer.

As for instance, even if Diaspora claims to ensure the Backward Right Revocation property, the removed user could be the pod administrator. Indeed, My3, RetroShare, Friendica, avoid to ensure the Backward Right Revocation because the removed user could belong to the set of trusted replica peers. As a result, the BPR property can be guaranteed only by re-allocating the contents already published on the peer of the users that can access them.

Table 8
Property ensured by alternative solutions to cryptography-based DOSNs.

DOSN	Grant access BW	Deny access BRP
Diaspora [9]	✗	✓
Vis-a-Vis [37]	✗	–
My3 [59]	✗	✗
eXO [40]	✗	✓
DiDuSoNet [42]	✗	–
Friendica [10]	✗	✗
RetroShare [11]	✗	✗

Instead, in eXO the Backward Right Revocation property can be directly enforced by the content owner because contents are stored on his peer.

Via-a-Vis stores unencrypted contents on the Virtual Identification Servers (VISs) trusted by the users and the Backward Right Revocation depends on the VISs while DiDuSoNet does not specify whether the Backward Right Revocation property is ensured or not.

Finally, the approach in [30,29] proposes to select the replica peers by choosing the ones belonging to users who are allowed to access the contents according to the related privacy policy. For these purposes, the content owner specifies a privacy policy for each content, describing the users who are authorized to access them by using privacy policy based on attributes (or features) derived from the user's profile. The privacy policy defined on each content is used to choose the set of trusted replica peers where to store unencrypted copy of the contents. In this case, every time a user wants to remove (or add) an authorized user, the proposed approach avoids any encryption operation. However, in the case of the removal of a user u from A, the allocation of the contents on the peers should be updated to guarantee the Backward Right Revocation property because the contents in C could have been allocated on the peer of u.

7. Discussion

This section discusses the implications of the security mechanisms adopted by the current DOSNs on the privacy level they guarantee to their users. In particular, based on the observations made in the qualitative analysis presented in the previous sections, we identify a set of properties which are relevant to assess the expressiveness of the privacy support of DOSNs. These properties are related to the ways a DOSN grant their users the capability to define privacy policies, i.e., respectively, to the possibility to define privacy polices based on Groups, Relationships, Profiles, and Contents.

In Table 9 we summarize the privacy policies supported by each of the DOSNs we examined. The Access Tracking property refers to the capability of the DOSNs to hide the access patterns to contents. Instead, in Table 10 we focus both on the enforcement of the policies and on the performance properties of the privacy support, namely: use of encryption for protecting contents (Encryption), selection of replicas based on trust among users (Trust replication),

Table 9
Summary of the privacy properties provided by current DOSNs. A property is marked ✓ if the DOSN support it.

Class	DOSN	Privacy Model				Access tracking
		Group	Relationship	Profile	Content	
Decentralized	LifeSocial.KOM	✓	✓			
	RetroShare	✓	✓			✓
	PeerSoN	✓	✓			
	DECENT		✓			
	LotusNet		✓		✓	✓
	eXO		✓			
	Cachet		✓			✓
Semi Decentralized	Soup	✓	✓	✓		
	Friendica	✓	✓	✓		
	Prometheus		✓	✓	✓	
	Safebook	✓	✓		✓	✓
	Gemstone		✓	✓		
	ProofBook	✓	✓			
	SuperNova		✓			
	DiDuSoNet		✓			
	Diaspora	✓				
	My3					
Hybrid	Contrail	✓		✓	✓	
	Persona	✓	✓			
	Vis-a-Vis	✓	✓			
	SocialGate		✓			
	Vegas		✓			

and the performance achieved by the join (Join), the leave (Leave), and initialization (Init) operations. In addition, we specify whether join and leave operations support respectively the Backward Secrecy property (BW) and the Backward Right Revocation property (BRP), because these can affect the performance of the operations.

The first interesting result that emerges from Table 9 is that none of the privacy supports of current DOSNs implements the all the properties previously described. In other words, in order to enable their users to protect their personal content, current DOSNs provide simple privacy models, where users are enabled to specify privacy policies mainly based on their relationships. The solutions implemented by LifeSocial.KOM, RetroShare, and PeerSoN suffer of a privacy model providing very basic access control capabilities that allow users to define privacy policies only based on groups and relationships. Also hybrid solutions, such as the ones adopted by Persona or Vis-a-Vis, provide the same properties of decentralized approaches, except for Contrail, which support privacy policies based on group, profile, and content information. However, their main issue is that they depend on some centralized entity (such as cloud provider or web service provider) that users have to trust in order to use the service. Moreover, our analysis reveals that semi-decentralized solutions extend the privacy model by providing also privacy policies based on either profile (e.g., Soup and Friendica) or content (e.g., Safebook). Instead, in order to make easier for users to express their privacy preferences Prometheus support the definition of privacy policies based on relationship, profile, and content but it does not allow its users to define privacy policy based on group.

Relationships-based properties provided by almost all current DOSNs are limited and very simple because they allow users to choose among a set of predefined access control options, mainly based on the friends' identities or on the types of the relationships. However, these simple privacy models suffer from several drawbacks. In fact, in addition to the type, a relation may also have a set of attributes that model properties and characteristics of the relationship (such as trust or strength, location of the relationship). In addition, besides friendship relations, also relationships between users and resources (such as owner and co-owner) should be exploited by DOSN's users to define privacy policies that take advantage of this information. As a result, the privacy models of current DOSNs may not be sufficient for supporting the privacy needs of the users and the definition of models enabling the definition of more granular policies is required.

While the privacy models of today's DOSNs fall short in providing full support to the privacy properties previously listed, the privacy of users with respect to their replica peers is typically overlooked. In particular, most of the current DOSNs exploit encryption to prevent replica peers from collecting private information of users. In addition, replication increases the possibility to hide the access patterns to the content, because the requester can choose an arbitrary replica to download it. However, this solution is useless if the identity of users, the relationship between them, as well as their actions are disclosed to third untrusted peers of the DOSNs. As shown in the Table, only a few DOSNs provide more advanced approaches to avoid access tracking. Safebook provides a Trusted Identification Service (TIS) that assures to each user at most one unambiguous identifiers and proposes to use onion routing technique in order to provide anonymity of the users. A similar solution is adopted by LotusNet, and Cachet where the identity of a user is associated with a random identifier and interaction among members can be concealed either with anonymous communication channels (such as Tor) or with private information retrieval protocols. In contrast to the previous DOSNs, LifeSocial.KOM, DECENT, Diaspora, PeerSoN, SuperNova, Vis-a-Vis, Persona, eXO, DiDuSoNet do not provide strong anonymity protection against access tracking, even if they can be extended to support anonymous communication.

By investigating the DOSNs exploiting other strategies than encryption to enforce the privacy preferences, we observed that the most part of them require that users explicitly select the user's peers where to store their contents. Instead, only a few existing DOSNs perform automatic selection of the replica peers based on either trust model (such as DiDuSoNet) or privacy policy defined by users. In addition, DOSNs that avoid encryption of the contents and that leverage replication (such as, eXO, DiDuSoNet, My3, and Friendica) are forced to share with the replica peers some private information of the users, such as the identities of the users who are authorized to access the contents.

Another important challenge of current DOSNs is guaranteeing a privacy support characterized by a good performance, in order to avoid that it negatively affects the user experience. The comparison of the proposed approaches shown in Table 10 provides important insights concerning their practical utilization. Indeed,

Table 10
A summary of the performance properties provided by privacy mechanisms of current DOSNs. A property is marked ✓ in the table if the DOSN support it while the performance of privacy mechanisms ranges between low (●) and high (●●●), based on time complexity shown in Tables 4–6.

Class	DOSN	Encryption	Trust replication	Join	BW	Leave	BRP	Init
Decentralized	LifeSocial.KOM	✓	✓	●		●	✓	●
	RetroShare			●●●		●●●		●●●
	PeerSoN	✓	✓	●		●	✓	●
	DECENT	✓	✓	●●●		●●●	✓	●●
	LotusNet	✓	✓	●●●		●●●		●
	eXO		✓	●●●		●●●	✓	●●●
	Cachet	✓	✓	●●●		●●	✓	●●
Semi Decentralized	Soup	✓	✓	●●●		●●	✓	●●
	Friendica		✓	●●●		●●●		●●●
	Prometheus	✓	✓	●●		●		●
	Safebook	✓	✓	●●		●		●●
	Gemstone	✓	✓	●●●		●●	✓	●●
	ProofBook	✓	✓	●●		●		●
	SuperNova	✓	✓	●●●		●●●		●●●
	DiDuSoNet		✓	●●●		●●●		●●●
	Diaspora			●●●		●●●	✓	●●●
Hybrid	My3		✓	●●●		●●●		●●●
	Contrail	✓		●●●	✓	●●●		●●
	Persona	✓	✓	●●●		●●	✓	●●
	Vis-a-Vis			●●●		●●●		●●●
	SocialGate	✓		●●●		●●●		●●
	Vegas	✓		●		●	✓	●

privacy mechanisms with higher performance level are desirable because they reduce as much as possible the impact that enforcement of the privacy policies has on the performance of the system (overhead, memory usage, bandwidth). We observed that the most part of current DOSNs are able to provide a simple and efficient implementation of the join operation. In fact, only LifeSocial.KOM, PeerSoN, and Vegas have low performance. However, most of them do not consider important aspects of the join operation because they do not deny the accesses of the newly authorized users to the contents previously published (i.e., they do not guarantee the BW property). The leave operations, instead, has low performance on 6 out of 22 DOSNs, and only 3 of these guarantee the BRP property. Finally, the initialization operation has low performance on 6 out of 22 DOSNs (5 of which are the ones having low performances on the leave operation). We observed that the approaches exploited by current DOSNs may not be suitable for implementing some privacy policies required by users. As for instance, the solutions characterized by low performance are not suitable for managing the accesses of large number of authorized users, while privacy mechanisms having higher performance can be exploited for managing large number of users, even in the presence of a high number of addition and removal of users. As a result the design of current DOSNs can be further improved by adopting different privacy mechanisms for enforcing different privacy policies. Among the promising approaches that can be adapted to the DOSN scenario in order to increase privacy of users we have: (i) Dynamic Identity-based Broadcast Encryption (DIBBE) [22], that allows the distribution of encrypted contents to a dynamic set of users, based on their identities; (ii) Dynamic Group Key Agreement (GKA) [80] where distributed algorithm is used to establish a common secret key between authorized users, and (ii) Hierarchical Key Assignment [78] where hierarchical structures formed by a certain number of authorized members are used to distribute contents. Moreover, some of these approaches can be easily adapted to provide the BW property for the user join and the BRP property for the user leave operation.

Finally, we notice that all the DOSNs we surveyed based their privacy models on the traditional access control schema, and none of them adopted advanced authorization supports such as the one presented in [81] implementing the Usage Control model [82].

8. Conclusion

In this paper we investigated the privacy mechanisms provided by the existing DOSNs in order to protect the privacy of the contents published by their users.

We selected a relevant number of DOSNs and we investigated the mechanisms they provide to allow users to express their privacy preferences, i.e., to decide which of the contents they published should be disclosed to the other users. In particular, we classified and compared the different types of supports for expressing privacy policies provided to the users to specify access rights to the contents of their profiles.

Moreover, we investigated the mechanisms adopted by these DOSNs in order to ensure that privacy policies defined by users are properly enforced. We found out that privacy policies are mainly enforced exploiting encryption, through a hybrid schema based on both symmetric and asymmetric cryptography. In addition, we observed that the security solutions exploited by DONS to enforce a privacy policy could be affected by the type of the privacy policy. As for instance, classical P2P security solutions could suffer from scalability issues if they are used for the enforcement of group-based privacy policies because the overhead introduced by encryption operations in order manage very large groups.

We investigated better the above problem by measuring the overhead introduced by privacy policy management (i.e., initialization and modification of a privacy policy) and by comparing the performance of each approach in terms of number of cryptographic keys created (#Key), and number of encryption operations required (#Enc). These analyses reveal that the most expensive operations are initialization of a privacy policy and removal of a user from the set of authorized member (which mainly depends on the number of members of the group).

Finally, we have also examined whether changes in privacy policies ensure (or not) the Backward Secrecy property and the Backward Right Revocation property. We noted that current DOSNs do not prevent a new user to access old contents published by the contents' owner, while users of the DOSNs could benefit from such property in the case they want to add a new member to a group without disclosing the contents already published in the group. Instead, in the case of user's removal, the Backward Right Revocation property is guaranteed only by some DOSNs. Furthermore, the proposed solutions lack of flexibility because the users cannot customize these properties for a specific user or group.

References

[1] N.B. Ellison, et al., Social network sites: Definition, history, and scholarship, J. Comput.-Mediat. Commun. 13 (1) (2007) 210–230.
[2] M. O'Connor, Facebook Revealed Private Email Addresses Last Night, GAWKER, 2010.
[3] G. Greenwald, E. MacAskill, NSA Prism program taps in to user data of Apple, Google and others, Guardian 7 (6) (2013) 1–43.
[4] E. Steel, G. Fowler, Facebook in privacy breach, Wall Str. J. 18 (2010).
[5] C.-m.A. Yeung, I. Liccardi, K. Lu, O. Seneviratne, T. Berners-Lee, Decentralization: The future of online social networking, in: W3C Workshop on the Future of Social Networking Position Papers, Vol. 2, 2009, pp. 2–7.
[6] A. Datta, S. Buchegger, L.-H. Vu, T. Strufe, K. Rzadca, Decentralized online social networks, in: Handbook of Social Network Technologies and Applications, Springer, 2010, pp. 349–378.
[7] I.A. Klampanos, J.M. Jose, Searching in peer-to-peer networks, Comput. Sci. Rev. 6 (4) (2012) 161–183.
[8] C. Selvaraj, S. Anand, A survey on security issues of reputation management systems for peer-to-peer networks, Comput. Sci. Rev. 6 (4) (2012) 145–160.
[9] R.S.D. GRippi, M. Salzberg, I. Zhitomirskiy, DIASPORA*. https://joindiaspora.com/.
[10] Friendica. http://friendi.ca/.
[11] RetroShare. http://retroshare.sourceforge.net/.
[12] B. Greschbach, G. Kreitz, S. Buchegger, The devil is in the metadata — new privacy challenges in decentralised online social networks, in: 2012 IEEE International Conference on Pervasive Computing and Communications Workshops, in: PERCOM Workshops, IEEE, 2012, pp. 333–339.
[13] S. Rathore, P.K. Sharma, V. Loia, Y.-S. Jeong, J.H. Park, Social network security: Issues, challenges, threats, and solutions, Inform. Sci. 421 (2017) 43–69.
[14] T. Paul, A. Famulari, T. Strufe, A survey on decentralized online social networks, Comput. Netw. 75 (2014) 437–452.
[15] S. Taheri-Boshrooyeh, A. Küpçü, Ö. Özkasap, Security and privacy of distributed online social networks, in: 2015 IEEE 35th International Conference on Distributed Computing Systems Workshops, ICDCSW, IEEE, 2015, pp. 112–119.
[16] D. Koll, J. Li, X. Fu, The good left undone: Advances and challenges in decentralizing online social networks, Comput. Commun. (2017).
[17] A. Sattikar, D.R. Kulkarni, A review of security and privacy issues in social networking, Int. J. Comput. Sci. Inf. Technol. 2 (6) (2011) 2784–2787.
[18] L. Schwittmann, M. Wander, C. Boelmann, T. Weis, Privacy preservation in decentralized online social networks, IEEE Internet Comput. 18 (2) (2014) 16–23.
[19] S.R. Chowdhury, A.R. Roy, M. Shaikh, K. Daudjee, A taxonomy of decentralized online social networks, Peer-to-Peer Netw. Appl. 8 (3) (2015) 367–383.
[20] 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: 2011 IEEE Consumer Communications and Networking Conference, IEEE, 2011, pp. 554–558.
[21] 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.
[22] O. Bodriagov, S. Buchegger, Encryption for peer-to-peer social networks, in: Security and Privacy in Social Networks, Springer, 2013, pp. 47–65.
[23] K. Graffi, P. Mukherjee, B. Menges, D. Hartung, A. Kovacevic, R. Steinmetz, Practical security in p2p-based social networks, in: IEEE 34th Conference on Local Computer Networks, LCN 2009, IEEE, 2009, pp. 269–272.
[24] Y. Challal, H. Seba, Group key management protocols: A novel taxonomy, Int. J. Inf. Technol. 2 (1) (2005) 105–118.
[25] H. Balakrishnan, M.F. Kaashoek, D. Karger, R. Morris, I. Stoica, Looking up data in P2P systems, Commun. ACM 46 (2) (2003) 43–48.
[26] Y.-K.R. Kwok, Peer-to-Peer Computing: Applications, Architecture, Protocols, and Challenges, CRC Press, 2011.
[27] S. Nilizadeh, S. Jahid, P. Mittal, N. Borisov, A. Kapadia, Cachet: a decentralized architecture for privacy preserving social networking with caching, in: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, ACM, 2012, pp. 337–348.
[28] A. De Salve, R. Di Pietro, P. Mori, L. Ricci, A logical key hierarchy based approach to preserve content privacy in decentralized online social networks, IEEE Trans. Dependable Secure Comput. PP (99) (2017) 1. http://dx.doi.org/10.1109/TDSC.2017.2729553.
[29] A. De Salve, P. Mori, L. Ricci, R. Al-Aaridhi, K. Graffi, Privacy-preserving data allocation in decentralized online social networks, in: Distributed Applications and Interoperable Systems, Springer, 2016, pp. 47–60.
[30] A. De Salve, P. Mori, L. Ricci, A privacy-aware framework for decentralized online social networks, in: International Conference on Database and Expert Systems Applications, Springer, 2015, pp. 479–490.
[31] L.A. 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.
[32] W.L. Ali, Securing safebook: Secure data access control and key management for safebook (Dissertation), 2013. Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-126987.
[33] Y. Afify, Access Control in a Peer-To-Peer Social Network (Master's thesis), EPFL, Lausanne, Switzerland, 2008.
[34] L.M. Aiello, G. Ruffo, LotusNet: tunable privacy for distributed online social network services, Comput. Commun. 35 (1) (2012) 75–88.
[35] R. Sharma, A. Datta, Supernova: Super-peers based architecture for decentralized online social networks, in: 2012 Fourth International Conference on Communication Systems and Networks, IEEE, 2012, pp. 1–10.
[36] R.S. Sandhu, P. Samarati, Access control: principle and practice, IEEE Commun. Mag. 32 (9) (1994) 40–48.
[37] A. Shakimov, H. Lim, R. Cáceres, L.P. Cox, K. Li, D. Liu, A. Varshavsky, Vis-a-vis: Privacy-preserving online social networking via virtual individual servers, in: 2011 Third International Conference on Communication Systems and Networks, IEEE, 2011, pp. 1–10.
[38] R. Narendula, T.G. Papaioannou, K. Aberer, A decentralized online social network with efficient user-driven replication, in: 2012 International Conference on Privacy, Security, Risk and Trust (PASSAT), and 2012 International Confernece on Social Computing (SocialCom), IEEE, 2012, pp. 166–175.
[39] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, D. Starin, Persona: an online social network with user-defined privacy, ACM SIGCOMM Comput. Commun. Rev. 39 (4) (2009) 135–146.
[40] A. Loupasakis, N. Ntarmos, P. Triantafillou, D. Makreshanski, eXO: Decentralized autonomous scalable social networking, in: CIDR, 2011, pp. 85–95.
[41] M. Durr, M. Maier, F. Dorfmeister, Vegas–a secure and privacy-preserving peer-to-peer online social network, in: 2012 International Conference on Privacy, Security, Risk and Trust (PASSAT), and 2012 International Confernece on Social Computing (SocialCom), IEEE, 2012, pp. 868–874.
[42] B. Guidi, T. Amft, A. De Salve, K. Graffi, L. Ricci, DiDuSoNet: A P2P architecture for distributed Dunbar-based social networks, Peer-to-Peer Netw. Appl. (2015) 1–18.
[43] F. Tegeler, D. Koll, X. Fu, Gemstone: empowering decentralized social networking with high data availability, in: 2011 IEEE Global Telecommunications Conference, GLOBECOM 2011, IEEE, 2011, pp. 1–6.
[44] RetroShare official documentation. https://retroshare.readthedocs.io.
[45] D. Koll, J. Li, X. Fu, Soup: an online social network by the people, for the people, in: Proceedings of the 15th International Middleware Conference, ACM, 2014, pp. 193–204.
[46] S. Biedermann, N.P. Karvelas, S. Katzenbeisser, T. Strufe, A. Peter, ProofBook: An online social network based on proof-of-work and friend-propagation, in: SOFSEM, Springer, 2014, pp. 114–125.
[47] S. Jahid, S. Nilizadeh, P. Mittal, N. Borisov, A. Kapadia, DECENT: A decentralized architecture for enforcing privacy in online social networks, in: 2012 IEEE International Conference on Pervasive Computing and Communications Workshops, IEEE, 2012, pp. 326–332.
[48] D. Koll, D. Lechler, X. Fu, SocialGate: Managing large-scale social data on home gateways, in: 2017 IEEE 25th International Conference on Network Protocols, ICNP, IEEE, 2017, pp. 1–6.
[49] S.R. Kruk, S. Grzonkowski, A. Gzella, T. Woroniecki, H.-C. Choi, D-FOAF: Distributed identity management with access rights delegation, in: The Semantic Web, ASWC 2006, Springer, 2006, pp. 140–154.
[50] H.C. Choi, S.R. Kruk, S. Grzonkowski, B. Davis, J. Breslin, Trust models for community aware identity management, 2006.
[51] A. Tootoonchian, S. Saroiu, Y. Ganjali, A. Wolman, Lockr: better privacy for social networks, in: Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, ACM, 2009, pp. 169–180.
[52] B. Ali, W. Villegas, M. Maheswaran, A trust based approach for protecting user data in social networks, in: Proceedings of the 2007 Conference of the Center for Advanced Studies on Collaborative Research, IBM Corp., 2007, pp. 288–293.
[53] B. Carminati, E. Ferrari, A. Perego, Enforcing access control in web-based social networks, ACM Trans. Inf. Syst. Secur. 13 (1) (2009) 6.
[54] A.C. Squicciarini, M. Shehab, F. Paci, Collective privacy management in social networks, in: Proceedings of the 18th International Conference on World Wide Web, ACM, 2009, pp. 521–530.
[55] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, B. Thuraisingham, Semantic web-based social network access control, Comput. Secur. 30 (2) (2011) 108–115.
[56] R. Nasim, S. Buchegger, XACML-based access control for decentralized online social networks, in: Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, IEEE Computer Society, 2014, pp. 671–676.
[57] OASIS, XACML version 3.0, 2013. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html.
[58] H. Hu, G.-J. Ahn, J. Jorgensen, Detecting and resolving privacy conflicts for collaborative data sharing in online social networks, in: Proceedings of the 27th Annual Computer Security Applications Conference, ACM, 2011, pp. 103–112.

[59] R. Narendula, T.G. Papaioannou, K. Aberer, My3: A highly-available P2P-based online social network, in: 2011 IEEE International Conference on Peer-to-Peer Computing, IEEE, 2011, pp. 166–167.
[60] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: IEEE Symposium on Security and Privacy, SP'07, IEEE, 2007, pp. 321–334.
[61] Diaspora Foundation. https://wiki.diasporafoundation.org/FAQ_for_users.
[62] L.A. Cutillo, M. Manulis, T. Strufe, Security and privacy in online social networks, in: Handbook of Social Network Technologies and Applications, Springer, 2010, pp. 497–522.
[63] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, K. Fu, Plutus: Scalable secure file sharing on untrusted storage, in: Fast, Vol. 3, 2003, pp. 29–42.
[64] L.-H. Vu, K. Aberer, S. Buchegger, A. Datta, Enabling secure secret sharing in distributed online social networks, in: Annual Computer Security Applications Conference, ACSAC'09, IEEE, 2009, pp. 419–428.
[65] S. Jahid, P. Mittal, N. Borisov, EASiER: Encryption-based access control in social networks with efficient revocation, in: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ACM, 2011, pp. 411–415.
[66] O. Bodriagov, G. Kreitz, S. Buchegger, Access control in decentralized online social networks: Applying a policy-hiding cryptographic scheme and evaluating its performance, in: 2014 IEEE International Conference on Pervasive Computing and Communications Workshops, in: PERCOM Workshops, IEEE, 2014, pp. 622–628.
[67] J. Katz, A. Sahai, B. Waters, Predicate encryption supporting disjunctions, polynomial equations, and inner products, J. Cryptol. 26 (2) (2013) 191–224.
[68] A. De Salve, M. Dondio, B. Guidi, L. Ricci, The impact of users availability on on-line ego networks: a Facebook analysis, Comput. Commun. 73 (2016) 211–218.
[69] S.G. Roberts, R.I. Dunbar, T.V. Pollet, T. Kuppens, Exploring variation in active network size: Constraints and ego characteristics, Social Networks 31 (2) (2009) 138–146.
[70] 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 ACM/IFIP/USENIX 11th International Conference on Middleware, Springer-Verlag, 2010, pp. 212–231.
[71] RetroShare team blog. https://retroshareteam.wordpress.com/.
[72] P. Stuedi, I. Mohomed, M. Balakrishnan, Z.M. Mao, V. Ramasubramanian, D. Terry, T. Wobber, Contrail: Enabling decentralized social networks on smartphones, in: Proceedings of the 12th International Middleware Conference, International Federation for Information Processing, 2011, pp. 40–59.
[73] Friendica helpers. https://helpers.pyxis.uberspace.de/help/Groups-and-Privacy.
[74] R.I. Dunbar, The social brain: mind, language, and society in evolutionary perspective, Annu. Rev. Anthropol. 32 (1) (2003) 163–181.
[75] P. Stuedi, I. Mohomed, M. Balakrishnan, Z.M. Mao, V. Ramasubramanian, D. Terry, T. Wobber, Contrail: Decentralized and privacy-preserving social networks on smartphones, IEEE Internet Comput. 18 (5) (2014) 44–51.
[76] D.-W. Kwak, J. Kim, A decentralized group key management scheme for the decentralized P2P environment, IEEE Commun. Lett. 11 (6) (2007) 555–557.
[77] A. De Salve, R. Di Pietro, P. Mori, L. Ricci, Logical key hierarchy for groups management in distributed online social network, in: 2016 IEEE Symposium on Computers and Communication, ISCC, IEEE, 2016, pp. 710–717.
[78] A. Castiglione, A. De Santis, B. Masucci, F. Palmieri, A. Castiglione, X. Huang, Cryptographic hierarchical access control for dynamic structures, IEEE Trans. Inf. Forensics Secur. 11 (10) (2016) 2349–2364.
[79] C.K. Wong, M. Gouda, S.S. Lam, Secure group communications using key graphs, IEEE/ACM Trans. Netw. 8 (1) (2000) 16–30.
[80] Q. Wu, B. Qin, L. Zhang, J. Domingo-Ferrer, O. Farràs, J.A. Manjon, Contributory broadcast encryption with efficient encryption and short ciphertexts, IEEE Trans. Comput. 65 (2) (2016) 466–479.
[81] A. Lazouski, F. Martinelli, P. Mori, A prototype for enforcing usage control policies based on XACML, in: Trust, Privacy and Security in Digital Business, TrustBus 2012, in: Lecture Notes in Computer Science, vol. 7449, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 79–92.
[82] J. Park, R. Sandhu, The $UCON_{ABC}$ usage control model, ACM Trans. Inf. Syst. Secur. 7 (1) (2004) 128–174.