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

Abstract

Decentralized Online Social Networks (DOSNs) have recently captured the interest of users because of the more control given to the end users over their shared social contents. Indeed, user privacy issues related to the centralized Online Social Network (OSN) services (such as Facebook or Google+) does 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 DOSNs and compare them by exploiting several criteria. After presenting the differences that the proposed DOSNs present in terms of provided services/architecture, we identify, for each DOSN, the privacy model used to define the privacy policies and the mechanisms for the management of the privacy policy (i.e., initialization and modification of the privacy policy). In addition, we evaluate the overhead introduced by security mechanisms adopted for privacy policy managements by discussing their advantages and drawbacks.

Keywords: Data privacy, Decentralized Online Social Network, Access Control, Security, Peer to peer computing

1. Introduction

Recent years have seen unprecedented growth in the Online Social Network (OSN) services [1], with about 300 OSNs collecting information about more than half a billion registered users[1]. An OSN enables its users to define their own *profile*, a virtual representation of themselves containing personal information,

*Corresponding author
Email addresses: andrea.desalve@iit.cnr.it (Andrea De Salve), paolo.mori@iit.cnr.it (Paolo Mori), ricci@di.unipi.it (Laura Ricci)

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

photos, posts and to explicitly declare the relationships with the (profiles of the) other users. Regardless of their purpose, the main service provided by OSNs to their users is the sharing of information with a set of selected contacts. Users can publish very heterogeneous contents, ranging from personal information, wall posts, photos, videos, comments to other posts, and they can send private messages.
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).

In order to use centralized OSN, users have to accept the related policy, which gives to the provider the right to access and to use users' data for several purposes, even though this data is intended for users' friends only. For instance, the Terms Of Service (TOS) established by Facebook allows users to delete their accounts from the social network at any time, but Facebook still has the right to access and exploit the related contents. In particular, the information produced by users of the centralized OSNs are typically exploited by the service provider for targeted advertising, product recommendations and any kind of forecasts. However, such information are usually intended for a specific audience only (the friends of the users or a subset of them).

Due to the centralized infrastructures, users of the current OSNs are forced to share the information directed to their friends by means of the 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].

Regardless of their architectures, one of the main features provided by current OSNs is the capability of the users to define privacy preferences on the contents of their profiles, i.e., to define which other users are allowed to see their contents. The current privacy mechanisms provided by centralized OSNs are very simple and allow users to choose only from a limited set of privacy options for a content [5]. In fact, the lack of privacy mechanisms with a suitable granularity level and flexibility could lead to the disclosure of improper information to the different parts of the system 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 OSNs, are constantly increasing, members of OSNs need an effective way to define authorizations that protect their contents. In contrast, the business model of the centralized OSN is based on the capability of attracting a large number of users and of encouraging them to share a large number of contents with other users of the system [6, 7]. In several recent cases, providers of centralized OSNs have been blamed for giving to end users a false sense of control over their data privacy, thus introducing several privacy issues exposing users to a number of risks [8, 9]. As a matter of fact, data of the OSNs

were used by malicious users to infer personal information [10, 11] or to perform other harmful activities [12].

1.1. Motivations

Several studies show that privacy is an ever-increasing concern for OSNs' users [13] and, in many cases, the strength of privacy mechanisms for protecting contents is crucial for the success of the service [14]. Behind these, centralized OSN infrastructures have also raised several drawbacks [14] such as the dependency on a single service provider, no re-use of data coming from other OSNs [15], the availability of the service or scalability. However, the lack of privacy with respect the centralized provider is one of the main concerns that has driven both scientists and the open source community towards the development of alternative OSN platforms able to shift the control over data to the end-users.

To address the previous privacy issues and leave to the users the control on their data, researchers have proposed to decentralize the functionalities of the OSN, by implementing them in a distributed way. The resulting platform is known as Decentralized Online Social Network (DOSN) [15, 14] and it is 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 (corresponding to the users' devices) that store the contents and execute the tasks needed to provide a seamless service (such as, search for data [16], recommendation [17], etc..).
For instance, Diaspora [18] 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 [19], another popular DOSN based on a extensible plug-in architecture, which currently has more than 1,100 users. Finally, RetroShare [20] is a DOSN which exploits Friend-to-Friend network to manage and to store users' data.
Therefore, the proposed DOSNs allow to shift the control over data to the end user because contents generated by the users are stored on their devices and they are not controlled by a single OSN provider. However, the decentralization of the OSN service introduces several issues related to the availability of users' contents and their privacy with respect the other users of the system.

As for the case of the centralized OSN, the users of DOSNs are able to define their specific privacy policies on the contents they share in order to specify who should have the capability to access and alter these contents. User's contents must be protected from unauthorized access and only users who have the permission of the owner can access the contents. However, while users' content may be stored on their private devices until they are online, when they are off line, their contents must be kept available on other devices available in the system, and this requires the usage of proper strategies to prevent unauthorized access to such contents. Indeed, the most part of existing DOSNs try to address these

privacy concerns by combining different encryption schemes. To achieve fine-grained access control, each content generated by a user of the DOSN should be encrypted (separately for different sets of recipients) before being stored and replicated on different users' devices. As a consequence, the design of DOSNs may suffer from performance issues that arise due to the overhead introduced by cryptographic schemes. For instance, authors of [21, 22, 23] investigated the overhead introduced by encryption schemes used by some DOSNs and highlighted the impact they have on performance and user experience. 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 modify them.

1.2. Contributions

The main aim of this paper is the investigation of the different approaches used by current DOSNs to protect the privacy of the contents of their users by analyzing and comparing them. For this reason, we identified a large set of DOSNs which have been proposed in the literature, without focusing only on the DOSNs which are really deployed and under active development (such as Diaspora, Friendica, or RetroShare). For each of the selected DOSN, first we briefly analyze the architectural model used to provide independence from a centralized provider and then we present the approaches implemented to enable users to define their privacy preferences (i.e., the privacy model and the privacy policy management). In particular, the privacy model is related to the capability of users to protect their contents by defining privacy policies that specify the set of authorized users. To help the reader in understanding the characteristics of the different privacy models, we classify them by using a taxonomy based on attributes.

Since the proposed DOSNs enforce the access control requirement by implementing their own protocol we investigate the approaches proposed for privacy policy management. In particular, we focus on the solutions exploited to guarantee that the privacy policies created by DOSNs' users are properly enforced on each content by using proper security mechanisms. In addition, we investigate the approaches used to enable the modification of the privileges on the contents, i.e., granting access to new users or revoking access to previously authorized users. We evaluate the overhead introduced by the privacy policy management in terms of number of cryptographic keys created, and number of encryption operations required. Finally, we have also determined whether changes in privacy policies ensure (or not) the backward secrecy property [24]. Besides presenting a comprehensive comparison of the privacy mechanisms used by the current DOSNs, this paper allows the reader to better understand the main privacy requirements of DOSNs, and how they impact the performances of the current DOSNs implementations.

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 and discuss the limitations and advantages of the current DOSNs in Section 6 and draw the final remarks in Section 7.

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 [14] 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-decentalized: 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 to exploit permanently available resources which guarantee that the contents of the users 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 [27], 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.

Diaspora. Among the most popular DOSNs we mention Diaspora [18, 28], a hybrid architecture where users may agree to act as a local server in order to keep complete control of their data, or choose to use an existing server (named Pod) from a Federated Social Pods Network, which consists of 184 open pods managed by third parties, having uptime of about 99%[2]. Generally, running a Pod requires a lot of different components to be installed and the amount of resources required by the server depends on how many users and how much traffic the installation receives.

Safebook. Safebook [29] relies on both a structured P2P overlay (Kademlia) and an unstructured network layer, named Matryoshka: concentric circles of nodes built around each member's node providing trusted data storage, profile data retrieval, and communication obfuscation [29]. The DHT is used for data lookup while the Matryoshka overlay is used for data retrieval. The innermost ring of a matryoshka is composed of direct contacts and each user associates a particular trust level for each of their friends. This level is used to select closely related contacts that will store the user's data. Users in successive logical rings are logically connected by friendship or trust relationships. The outermost ring acts as a gateway and the peers in this ring route all the requests addressed to the central node towards the innermost ring of the Matryoshka. As a result, anonimity is guaranteed by the Matryoshka structure because the center peer storing data of a friend is not able to understand which is the user requesting such data. The profile of a user is logically organized as a hierarchy where contents are located only at the leaves of the profile hierarchy tree [30].

PeerSoN. The architecture of PeerSoN [31] is originally designed in [32]. The PeerSoN exploits both a DHT to implement a lookup service and also direct communications between the peer of a user and those of his friends. Users store their information on their local devices and they directly exchange information with their friends, if they are online. The structured P2P overlay is used as a lookup support to find a friend, to store the updates for the users which are offline, and to store peer's metadata, while contents are exchanged on-demand, via a direct communication between the involved peers.

SuperNova. SuperNova [33] is a super-peer based architecture which relies on an unstructured overlay. Each peer of the system can act as a *super-peer*, by providing and managing different types of services, or as *storekeeper*, by keeping a copy of the data of their friends. Users' data can be stored on unknown peers

[2] `https://podupti.me/`

(super-peers) or users' friends peers (storekeepers). Initially, since new users who join the system may not have enough friends in the network, they store their data on the super-peers. After this initial phase, the system tries to find storekeepers among the user's friends, in order to store data on them as well. Super-peers are responsible for friends lookup, finding potential storekeepers for friend, keeping track of replica and tracking the availability of the users assigned to them.

LifeSocial.KOM. LikeSocial.KOM [34] is a plugin-based architecture which exploits a structured P2P overlay (i.e., FreePastry). Users' contents are securely stored on the DHT and they can be quickly retrieved by using the key identifier of the content. In order to ensure availability against involuntary disconnections of users' peers, PAST [35] is used. In particular, each content is replicated on the k nodes having identifiers closest to the identifier of the content.

Cachet. Cachet [36] is based on DECENT [37]: an architecture for DOSNs that exploits a structured P2P (such as Pastry or Kademlia) to store user data. In particular, contents are securely stored on the DHT by using the identifier of the content as the DHT key. The data replication service of the underlying DHT is exploited to guarantee data availability. In addition to the underlying DHT, the architecture of Cachet also exploits a gossip-based social caching algorithm where the devices of the trusted friends are leveraged to provide cached, decrypted contents to other contacts. Contents are organized in container objects which may contain, in turn, references to other containers. Containers are the basic unit of access control.

LotusNet. LotusNet [38] is a DOSN based on a structured P2P overlay where users' data could be stored and replicated on distinct peers which build up the DHT. The DHT used by the system is a customized version of Kademlia (called Likir [38]), which enhances the Kademlia protocol by introducing identity management. The architecture can be easily extended by providing new features (or *widgets*) on top of the P2P structured layer, while communications between widgets takes place through the DHT.

Vis-à-Vis. Vis-à-Vis [39] leverages an hybrid architecture where each user is paired to an independent *Virtual Identification Server* (VIS) providing a reliable storage service which also guarantee high availability of user data. A VIS is a virtual machine provided as a cloud computing service by third parties which are external to the system. Users select and configure their VISs at the moment of registration, and users' data are stored on the selected VISs. The VIS of a user acts as a proxy for their contents, by providing them on behalf of the user. Finally, location-based groups have a central role in Vis-a-Vis because VISs are organized according to a hierarchical routing structure where VISs are located on the leaves of the tree and intermediate nodes represent geographic regions.

My3. The initial design of My3, (initially referred as *porkut*), was originally described in [40] and further refined in [41] and [42]. My3 is based on a hybrid architecture based on both structured and unstructured P2P architecture. Users' contents are stored on a set of trusted peers self-chosen by the users among their friends. The set of trusted peers are selected by considering the availability and performance of the users' devices. In addition, users' peers are organized according to a DHT, which is used for storing the mapping between users and the trusted peers where their contents are stored, in a privacy-preserving way.

Persona. Persona [43] leverages a hybrid architecture where users store their data either on their local storage service or on a dedicated web server provided by others external parties. The storage service is implemented through an API which provides to the users the possibility to store their contents and make them available to others users. Private server of private virtual space (such as Dropbox) can be used as external storage service.

eXO. Authors of [44] propose eXO, a DOSN based on a structured architecture, which provides advanced mechanisms for content/user discovery based on tags. Each peer belongs to a DHT (such as Pastry, Chord, and CAN) and contents are stored on the peer of the owner or on the DHT, for availability purposes. Each content is indexed by hashing a set of keywords that describe it and each user is paired to a user profile which is also indexed on the DHT.

Vegas. The authors of [45] propose Vegas, a DOSN hybrid architecture with limited capabilities and focused on mobility. Indeed, availability of the contents is guaranteed by exploiting external data storage services (such as FTP, WebDAV, Amazon S36, Google Drive, or Dropbox). To increase data decentralization, users are allowed to select several data stores which provide access to the contents published in their profiles. The storage service is responsible for the synchronization between multiple devices of the same user.

DiDuSoNet. Authors of [46] propose DiDuSoNet, a multilevel architecture consisting of an unstructured Dunbar Social Overlay and a structured DHT level. The first level is described in [47] and it provides a storage service which stores contents of a user only on a restricted number of friends' peers having regular contacts with him (named Dunbar friends [48]). The DHT provides the functionality to find users and to store, search, and retrieve contents in a distributed fashion. The DHT provides also bootstrap functionalities and support the search of new friends.

Prometheus. In [49], a P2P social-aware architecture, named Prometheus, is proposed, and it leverages a structured P2P infrastructure to provide the OSN services. In particular, it is based on the Mobius [50] architecture which consists of two layers: *i)* a mobile (wireless) user-centric layer which executes the mobile application, and *ii)* a structured P2P network (based on Pastry [51]) supporting the mobile applications on smartphones when they leave the mobile network. Indeed, the devices of the P2P layer have more powerful resources and they can

provide reliable services. Prometheus does not require each user to participate to the P2P network layer with his peer's resources. Instead, a user of the mobile layer can decide to exploit a social multimedia-sharing service provided by a trusted peer of the P2P network layer. The mobile application interacts with the trusted peer, which stores and replicates the contents shared by the user. Past [52] is used to ensure data availability by replicating contents produce by mobile applications on different peers.

Gemstone. Authors of [53] propose Gemstone: a DOSN platform based on a structured DHT, which only stores pointers to the contents, while contents are stored and replicated on distinct peers by using learning mechanisms. In particular, social relationships, online patterns of peers and user experiences are used to select the device where contents are stored. A content can be requested directly to the content owner (if online) or to replica peers. Indeed, when a user is online, his peer acts as a server by providing his profile and by ensuring consistency between the replica peers. When the user goes offline, the replicas are in charge of providing he profile of such user.

Friendica. The main purpose of the Friendica project [19] is the creation of a federated decentralized social platform, on the same line of the Diaspora project. Indeed, it is based on a hybrid architecture which exploits the same Diaspora federation network consisting of 309 nodes. However, only 52 Friendica nodes support the social network and they can be effectively used to host the users profiles. Alternately, if a user does not trust the administrator of a Friendica node, he can choose to use his own server to manage a new Friendica node. As stated from the online documentation in [19], the administrator of a Friendica node has information about the users profiles on his server, which can be moderated. Depending on the configuration of the administration, the Friendica nodes can decide to block or not public access to profiles, friends, photos, etc., to unauthorized persons unless they are logged in. Optionally, encrypted server-to-server communication can be used between the Friendica nodes who have appropriately configured this extension.

RetroShare. Another established DOSN is RetroShare [20], a decentralized Friend-to-Friend network (F2F) which consists of network of users' peers, where the exact location of the peer of each user is only known to his friends (or neighbors). Each user is paired with a public key which is used to identify people and to verify authenticity of the data. RetroShare allows direct secure connections only between two friends. As a result, data transfer between two users' peers who are not friends requires to search for a chain of friend-of-friend, connecting the sender to the recipient.
A user may choose to act as a relay for his friends, for the friends of his friends, or anyone in the RetroShare network. In such a case, the relay nodes behave as a bridge between the sender and the receiver by routing encrypted traffic in a tor-like style. A DHT based on BitTorrent is used only to locate friends and resources while data are stored on the private peers of the users. Shared

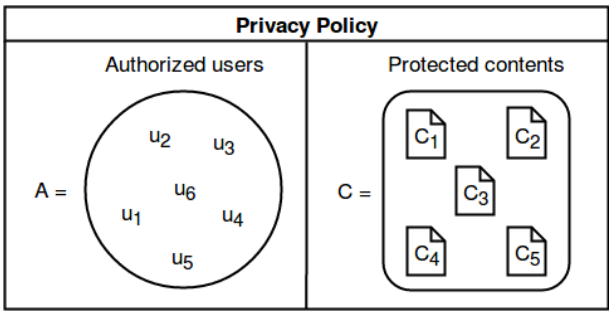

Figure 1: The elements of a privacy policy.

contents are hashed and become available for friends who can download them and make available to other users. As specified in the online documentation [20], the network availability improves as the number of friends grows and users have recommended to start with about 5 friends to make Retroshare worthwhile. Eventually, when not enough peers are available as trusted friends, a user can accept to expand their network by connecting also to friends of friends.

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. Figure 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.

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 [18], 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 owner 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.

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

DOSN	Enc	Encryption Schemes	Privacy Policy
Diaspora [18]	✗		private, public, selected contacts
Safebook [29]	✓	Asymmetric, Symmetric	private, group, attributes, trust level, depth
PeerSoN [31]	✓	Asymmetric, Symmetric	private, public, groups
LotusNet [38]	✓	Asymmetric, Symmetric	selected contacts, regular expression on content type
SuperNova [33]	✓	Asymmetric, Symmetric	private, public, selected contacts
LifeSocial.KOM [34]	✓	Asymmetric, Symmetric	private, public, selected contacts
Vis-a-Vis [39]	✗		group admission based on friendship and credentials
My3 [41]	✗		trusted contacts, all friends
Cachet [36]	✓	Asymmetric, Symmetric, ABE	identity or attribute-based policy
Persona (ABE) [43]	✓	Asymmetric, Symmetric, ABE	private, group, selected contacts, attribute-based group
eXO [44]	✗		public, private
Vegas [45]	✓	Asymmetric	selected contacts, all friends
DiDuSoNet [46]	✗		selected contacts, all contacts, dunbar circles
Prometheus [49]	✓	Asymmetric	relationship type, interactions, weights of the relationship, location
Gemstone [53]	✓	Symmetric, ABE	attribute-based policy
Friendica [19]	✗		public, selected groups, selected contacts
RetroShare [20]	✗		circles, selected groups, selected contacts, n-degree contacts

Safebook. The privacy model of Safebook is sketched in [29] and refined in [30]. 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 without encryption. We recall that mirrors are the innermost nodes of the Matryoshka.

PeerSoN. PeerSoN [54] 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 [38], 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 allows to maintain their number low, but does not allow the definition of granular privacy policies. 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 [33] 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 can not 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 [34] does not allow users to define complex privacy policies, but it provides a security layer [22] where users are enabled to define Access Control Lists (ACLs) [55] 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 [39] 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 [39]. 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 [42] 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 [36] 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 [43], 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 others. 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 [44], 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 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 can not be exploited during the definition of the privacy policies.

Vegas. Members of Vegas [45] 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 [46] 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 a small number of 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 [53] 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 [19] 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[3]. 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, a user can define a privacy policy that allows to share a content anonymously for untrusted friends but with the exception of the family group. The group owner selects a discovery policy for each group that specifies the group settings 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 [56] 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.

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 [57] 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 [58] extend the D-FOAF system by considering the case of multiple types of

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

relationships.
On the same line of research, the authors of [59] propose Lockr: a system exploiting relationships among users within the DOSN to specify privacy policy.

Authors in [60] 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 [61, 6] 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. [62] 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 [63, 23] proposed to exploit XACML [64] (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 [23] 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 [65] 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.

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 $A = \{a_1, \cdots, a_n\}$
n	the number of authorized users involved in a privacy policy
C	the set of contents $C = \{c_1, \cdots, c_m\}$ 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

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 [46, 41, 66], 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 Section 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 [67]). 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 $A = \{a_1, \cdots, a_n\}$ the permission to access the set of protected contents $C = \{c_1, \cdots, c_m\}$. 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 [68]. 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 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 [30]. 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 [69].

PeerSoN. PeerSoN [31, 54] 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[4] 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 to encrypt 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 [33] uses a cryptographic storage system [70] 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 [71] 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 [34, 22], 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

[4] `http://openid.net/`

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 [22] 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 milliseconds 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 [39] 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.

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) [42]. 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 [36], users' data are protected by EASiER [72]: a cryptographic hybrid structure where privacy policies $P(A, C)$ are enforced by using traditional public-private key and Attribute-Based Encryption (ABE) [67]. 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 don't satisfy the attributes specified by the ABE secret key in the privacy policy can't 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. [73] focus on the problem of ABE cryptographic primitives that hide the user's data but reveal access policies. They introduce predicate encryption (PE) [74] 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 [44]. 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 [45], 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 [46], a privacy policy $P(A, C)$ simply enables access the users's 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 [75, 76] of the relationships (in terms of amount of interactions).

Prometheus. Prometheus [49] 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 [53] 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 [19] 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 create. 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 shared immediately but after 2 minutes or up to 1 hour, 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 [77].

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 $P(A, C) \rightarrow P(A, C) + P(A \backslash \{w\}, C')$. 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 $P(A, C) \rightarrow P(A \backslash \{w\}, C) + P(A \backslash \{w\}, C')$ 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 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 [68]. 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 fully ensured 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 [29, 69], 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 [31, 54], 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. [71] 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 [38]. 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 PeerSoN 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 hasn't 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 members modify them. However, this solution is not effective because the majority of contents shared in OSNs are never modified [38].

SuperNova. Users of SuperNova [33] 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 [70]. 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 the his 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 [22]. 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 wants, 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 [39] 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 to enforce privacy policies by using encryption mechanisms and exploits trust friends for access control and storage of unencrypted contents [42]. 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 [36] 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 [72]. 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 can not 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 [44]. 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 [45]. 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 [46] 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 [49] 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 to securely share 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 [53], 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 olds 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 [78].

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. In a similar way, 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 an the contents in their local storage.

6. Evaluation and Discussion

The previous sections surveyed some crucial aspects of current DOSNs, and this section presents a comparison among them with respect to those aspects. Table 3 summarizes the architectural style of each DOSNs, while Table 4 describes mainly the characteristics of the storage layer. The architectural style adopted by the current DOSNs is mainly based on structured P2P architectures, such as OpenDHT in PeerSoN, FreePastry in LifeSocial.KOM, Likir in LotusNet, DECENT in Cachet, 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

Table 3: Evaluation of the architectural style of DOSNs.

	Routing		Overlay Network	
DOSN	**Structured**	**Unstructured**	**Structured**	**Unstructured**
Diaspora [18]		•		Federated Network
Safebook [29]	•	•	Kademlia	Matryoshka
PeerSoN [31]	•		OpenDHT	
LotusNet [38]	•		Likir	
SuperNova [33]		•		not specified
LifeSocial.KOM [34]	•		FreePastry	
Vis-a-Vis [39]		•		not specified
My3 [41]	•	•	DHT	not specified
Cachet [36]	•		DECENT	
Persona (ABE) [43]		•		Network of Private Servers
eXO [44]	•		DHT	
Vegas [45]		•		Network of Private Servers
DiDuSoNet [46]	•	•	DHT	Dunbar Social Overlay
Prometheus [49]	•	•	Pastry	Mesh network
Gemstone [53]	•	•	DHT	not specified
Friendica [19]		•		Federated Network
RetroShare [20]	•	•	BitTorrent	Mesh Network

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, 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 [36]. 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.

Behind these, there are also other 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 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 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, 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.
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.

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 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, 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. All 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, 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 DiDuSoNet, and [79, 66, 23]) adopt trust models to automatically derive these super peers.

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 Figure 2. In particular, we identified 4 different privacy models:

Table 4: Evaluation of the architectural style of the DOSNs' storage systems

DOSN	Data storage			Features	
	Decentralized	Semi-decentralized	Hybrid	Data Encryption	Replication
Diaspora [18]		•		✗	✗
Safebook [29]		•		✓	✓
PeerSoN [31]	•			✓	✗
LotusNet [38]	•			✓	✓
SuperNova [33]		•		✓	✓
LifeSocial.KOM [34]	•			✓	✓
Vis-a-Vis [39]			•	✗	✗
My3 [41]		•		✗	✓
Cachet [36]	•			✓	✓
Persona (ABE) [43]			•	✓	✓
eXO [44]	•			-	✓
Vegas [45]			•	✓	✗
DiDuSoNet [46]		•		✗	✓
Prometheus [49]		•		✓	✓
Gemstone [53]		•		✓	✓
Friendica [19]		•		✗	✗
RetroShare [20]	•			✗	✓

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 5, all the considered DOSNs except Diaspora, allow their users to define relationship-based privacy policies. Most of DOSNs, such as Safebook, Cachet, Persona, eXO, Vegas, DiDuSoNet, Prometheus, Gemstone, allow users to organize their contacts in homogeneous groups by specifying the type of a relationship (such as family, acquaintances, close friend, work 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 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, Friendica and Vegas to define privacy policies.

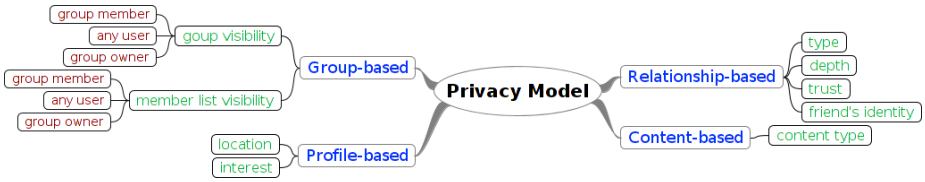

Figure 2: A taxonomy for the classification of privacy models.

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 [48]: 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 can contains contacts with different types of relationships. As a results, 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 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 [56], 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.

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 a content (such as Post, Comment, Like, or Photo) as attribute, and the privacy policies define access permissions based on content type. The type of a 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 and Gemstone can use attributes that model features originating from user-entered profile information (such as location and interest). These information 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 a 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, Persona, and Gemstone) that propose to improve the capabilities of existing approaches by integrating Attribute-based Encryption (ABE) [67]. In Section 6.2.2, instead, we describe some approaches that are not based on cryptography.

6.2.1. Cryptography-based DOSNs

Initialization. Table 6 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

Table 5: Classification of the privacy models provided by current DOSNs according to the taxonomy defined by Figure 2

.

DOSN	Group-based			Relationship-based	Profile-based	Content-based
	group name	group visibility	member list visibility			
Diaspora	aspect	group owner	group owner			
		group owner	group member			
Safebook	circles	group owner	group owner	type, depth, trust		content type
	public group	group member	group member			
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	group member	group member	friend's identity		
	public group	any user	any user			
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 groups	group owner	group owner	friend's identity	DNS location	
RetroShare	public groups	group owner	group owner	depth		
	public circle	any friend	anonymous			
	private circle	circle member	anonymous			
	restricted	circle member	anonymous			
	invited only	circle member	anonymous			

6 we don't 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.

Table 6: Evaluation of the overhead for privacy policy definition.

DOSN	Initialization	
	#Key	**#Enc**
Safebook [29]	$2 \cdot GenKeyS$	$m \cdot (2 \cdot EncS) + n \cdot EncAS$
PeerSoN [31]	$m \cdot GenKeyS$	$m \cdot (EncS + n \cdot EncAS)$
LotusNet [38]	$GenKeyS$	$m \cdot (EncS + n \cdot EncAS)$
SuperNova [33]	$GenKeyS$	$m \cdot EncS$
LifeSocial.KOM [34]	$m \cdot GenKeyS$	$m \cdot (EncS + n \cdot EncAS)$
Cachet [36]	$m \cdot GenKeyS$	$m \cdot (EncS + EncABE)$
Persona (ABE) [43]	$m \cdot GenKeyS$	$m \cdot (EncS + EncABE)$
Vegas [45]	$m \cdot GenKeyS$	$m \cdot (EncS + n \cdot EncAS)$
Prometheus [49]	GenKeyAS	$m \cdot EncAS + n \cdot EncAS$
Gemstone [53]	$m \cdot GenKeyS$	$m \cdot (EncS + EncABE)$

In order to ensure fine-grained and efficient access control, most of the current DOSNs (such as PeerSoN, LifeSocial.KOM, Cachet, 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, 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 encription 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. 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 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), 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

Table 7: Evaluation of the current approach for user addition.

DOSN	Grant access to a new user		
	#Key	#Enc	BW
Safebook [29]	0	$EncAS + EncS$	✗
PeerSoN [31]	0	$m \cdot EncAS$	✗
LotusNet [38]	0	0	✗
SuperNova [33]	0	0	✗
LifeSocial.KOM [34]	0	$m \cdot EncAS$	✗
Cachet [36]	$GenKeyABE$	0	✗
Persona (ABE) [43]	$GenKeyABE$	0	✗
Vegas [45]	0	$m \cdot EncAS$	✗
Prometheus [49]	0	EncAS	✗
Gemstone [53]	$GenKeyABE$	0	✗

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, Persona, 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. However, as mentioned by the authors of [43], this approach is affected by some performance penalty because ABE operations have proved to be about 100-1000 times slower than those of the RSA. 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 6. 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 7 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 all the current DOSNs do not guarantee the backward secrecy property (last column of Table 7, 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

Table 8: Evaluation of the current approach for the case of deny access.

DOSN	Deny access to a previously authorized user		
	#Key	#Enc	BRP
Safebook [29]	$GenKeyS$	$n \cdot EncS$	✗
PeerSoN [31]	$m \cdot GenKeyS$	$m \cdot (EncS + n \cdot EncAS)$	✓
LotusNet [38]	0	0	✗
SuperNova [33]	$GenKeyS$	0	✗
LifeSocial.KOM [34]	$m \cdot GenKeyS$	$m \cdot (EncS + n \cdot EncAS)$	✓
Cachet [36]	$m \cdot GenKeyS$	$m \cdot (EncS + EncABE)$	✓
Persona (ABE) [43]	$m \cdot GenKeyS$	$m \cdot (EncS + EncABE)$	✓
Vegas [45]	$m \cdot GenKeyS$	$m \cdot (EndS + n \cdot EncAS)$	✓
Prometheus [49]	$GenKeyAS$	$n \cdot EncAS$	✗
Gemstone [53]	$m \cdot GenKeyS$	$m \cdot (EncS + EncABE)$	✓

the contents already published in C. For this reason, the most part of current DOSNs (such as Safebook, PeerSoN, LifeSocial.KOM, Prometheus 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 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 futher encrypt this key pair. However, the estabilishment of the secure channel requires an additional cost which is not reported in Table 7. 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, 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.
It is interesting to note that it doesn't 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 8 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 8 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 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, Persona, Vegas, Gemstone) 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. However, 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 [21]. 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 [21, 22, 23] showed that some of the 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, 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 don't have a real user removal procedure, but they simply assign short validity periods to grant certificates and they don't 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 to use short duration certificates.

The second approach to mitigate the efficiency problem related to the removal of a user [80, 79, 81, 82] exploits the strength hierarchical data structures for reducing the overhead of the update of the privacy policy. As for

Table 9: Alternative solutions to cryptography-based DOSNs.

DOSN	Privacy Policy Management		
	User device	**Peers Selection**	**Scope**
Diaspora [18]	not required	trusted by users	internal (pods)
Vis-a-Vis [39]	not required	trusted by users	external (virtual Identification server)
My3 [41]	required	trusted by users	internal (trusted friends' peers)
eXO [44]	required	localhost	internal (localhost and DHT peers)
DiDuSoNet [46]	required	tie strength	internal (derived from trust model)
Friendica [19]	not required	trusted by users	internal (Friendica servers)
RetroShare [20]	required	localhost	internal

instance, authors of [80, 79, 81] of the Logical Key Hierarchy model (LKH) [83] 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 [79] 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 [80] propose a decentralized group key management algorithm which combines both the LKH and the tree-based group Diffie-Hellman (TGDH) where authorized members are manged 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 [82] propose to define hierarchy of groups where some subgroups have more or less 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.

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 9 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 10 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. 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 [23, 66] 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

Table 10: Property ensured by alternative solutions to cryptography-based DOSNs

DOSN	Grant access	Deny access
	BW	BRP
Diaspora [18]	✗	✓
Vis-a-Vis [39]	✗	-
My3 [41]	✗	✗
eXO [44]	✗	✓
DiDuSoNet [46]	✗	-
Friendica [19]	✗	✗
RetroShare [20]	✗	✗

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. 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 classified the architectural styles they adopted in order to accomplish their tasks, and the results of these analysis reveal that the most part of existing DOSNs exploit manly unstructured P2P architecture. Results of these analysis reveal that the most part of existing DOSNs exploit manly unstructured P2P architecture for routing (such as Diaspora, Friendica, Matryoshka in Safebook, Mesh in both Prometheus and RetroShare).

For each of the selected DOSN, we also 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 privacy policies provided to the users to specify access rights to the contents of their profiles. The results reveal that the friendships-based privacy policies provided by 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) can be exploited by DOSN's users to define privacy policies that take advantage of this information.

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 to enforce group-based privacy policies because the overhead introduced by encryption operations in order manage very large groups which can vary in size through the addition and removal of users.

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 member of the group). In order to mitigate such cost, some DOSNs (such as Diaspora, Friendica, and RetroShare) prefer to store unencrypted contents on the peers trusted by the contents' owner, by avoiding the use of cryptography for protecting the privacy of such contents.

We investigate in more detail the current DOSNs which exploit other strategies than encryption to enforce the privacy preferences of the users on their contents. We observed that the most part of DOSNs require that users have explicitly selected the user's peers where to store their contents and that users trust them. 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.

Finally, we have also summarized whether changes in privacy policies ensure (or not) the Backward Secrecy property and the Backward Right Revocation property. We noted that current DOSNs does not prevent a new user to access old contents published by the contents' owner. However, 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 typically guaranteed by current DOSNs because the removed users cannot still access old contents. However, the proposed solutions lack of flexibility because they are fixed and the users cannot customize these property for a specific user or group.

References

[1] N. B. Ellison, et al., Social network sites: Definition, history, and scholarship, Journal of Computer-Mediated Communication 13 (1) (2007) 210–230.

[2] M. O'Connor, Facebook revealed private email addresses last night (2010).

[3] G. Greenwald, E. MacAskill, Nsa prism program taps in to user data of apple, google and others, The Guardian 7 (6) (2013) 1–43.

[4] E. Steel, G. Fowler, Facebook in privacy breach, The Wall Street Journal 18.

[5] A. C. Squicciarini, M. Shehab, J. Wede, Privacy policies for shared content in social network sites, The VLDB Journal-The International Journal on Very Large Data Bases 19 (6) (2010) 777–796.

[6] 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.

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

[8] R. Gross, A. Acquisti, Information revelation and privacy in online social networks, in: Proceedings of the 2005 ACM workshop on Privacy in the electronic society, ACM, 2005, pp. 71–80.

[9] G. Hogben, Security issues and recommendations for online social networks, ENISA position paper 1 (2007) 1–36.

[10] J. Lindamood, R. Heatherly, M. Kantarcioglu, B. Thuraisingham, Inferring private information using social network data, in: Proceedings of the 18th international conference on World wide web, ACM, 2009, pp. 1145–1146.

[11] T. Pontes, G. Magno, M. Vasconcelos, A. Gupta, J. Almeida, P. Kumaraguru, V. Almeida, Beware of what you share: Inferring home location in social networks, in: Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on, IEEE, 2012, pp. 571–578.

[12] A. Acquisti, R. Gross, Predicting social security numbers from public data, Proceedings of the National academy of sciences 106 (27) (2009) 10975–10980.

[13] C. Magazine, Facebook & your privacy who sees the data you share on the biggest social network? (2012).

[14] 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.

[15] 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.

[16] I. A. Klampanos, J. M. Jose, Searching in peer-to-peer networks, Computer Science Review 6 (4) (2012) 161–183.

[17] C. Selvaraj, S. Anand, A survey on security issues of reputation management systems for peer-to-peer networks, Computer Science Review 6 (4) (2012) 145–160.

[18] R. S. D. GRippi, M. Salzberg, I. Zhitomirskiy, Diaspora*, https://joindiaspora.com/.

[19] Friendica, http://friendi.ca/.

[20] Retroshare, http://retroshare.sourceforge.net/.

[21] O. Bodriagov, S. Buchegger, Encryption for peer-to-peer social networks, in: Security and Privacy in Social Networks, Springer, 2013, pp. 47–65.

[22] K. Graffi, P. Mukherjee, B. Menges, D. Hartung, A. Kovacevic, R. Steinmetz, Practical security in p2p-based social networks, in: Local Computer Networks, 2009. LCN 2009. IEEE 34th Conference on, IEEE, 2009, pp. 269–272.

[23] 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.

[24] Y. Challal, H. Seba, Group key management protocols: A novel taxonomy, International journal of information technology 2 (1) (2005) 105–118.

[25] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica, Looking up data in p2p systems, Communications of the ACM 46 (2) (2003) 43–48.

[26] Y.-K. R. Kwok, Peer-to-Peer Computing: Applications, Architecture, Protocols, and Challenges, CRC Press, 2011.

[27] D. Koll, J. Li, X. Fu, The good left undone: Advances and challenges in decentralizing online social networks, Computer Communications.

[28] A. Bielenberg, L. Helm, A. Gentilucci, D. Stefanescu, H. Zhang, The growth of diaspora-a decentralized online social network in the wild, in: Computer Communications Workshops, 2012 IEEE Conference on, IEEE, 2012, pp. 13–18.

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

[30] W. L. Ali, Securing safebook: Secure data access control and key management for safebook (dissertation), Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-126987 (2013).

[31] 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.

[32] D. Schiöberg, A peer-to-peer infrastructure for social networks, Diplom thesis, TU Berlin, Berlin, Germany.

[33] R. Sharma, A. Datta, Supernova: Super-peers based architecture for decentralized online social networks, in: Communication Systems and Networks, 2012 Fourth International Conference on, IEEE, 2012, pp. 1–10.

[34] 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: Consumer Communications and Networking Conference, 2011 IEEE, IEEE, 2011, pp. 554–558.

[35] P. Druschel, A. Rowstron, Past: A large-scale, persistent peer-to-peer storage utility, in: Hot Topics in Operating Systems, 2001. Proceedings of the Eighth Workshop on, IEEE, 2001, pp. 75–80.

[36] 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.

[37] S. Jahid, S. Nilizadeh, P. Mittal, N. Borisov, A. Kapadia, Decent: A decentralized architecture for enforcing privacy in online social networks, in: Pervasive Computing and Communications Workshops, 2012 IEEE International Conference on, IEEE, 2012, pp. 326–332.

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

[39] 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: Communication Systems and Networks, 2011 Third International Conference on, IEEE, 2011, pp. 1–10.

[40] R. Narendula, T. G. Papaioannou, K. Aberer, Privacy-aware and highly-available osn profiles, in: Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE), 2010 19th IEEE International Workshop on, IEEE, 2010, pp. 211–216.

[41] R. Narendula, T. G. Papaioannou, K. Aberer, My3: A highly-available p2p-based online social network, in: Peer-to-Peer Computing, 2011 IEEE International Conference on, IEEE, 2011, pp. 166–167.

[42] R. Narendula, T. G. Papaioannou, K. Aberer, A decentralized online social network with efficient user-driven replication, in: Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom), IEEE, 2012, pp. 166–175.

[43] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, D. Starin, Persona: an online social network with user-defined privacy, ACM SIGCOMM Computer Communication Review 39 (4) (2009) 135–146.

[44] A. Loupasakis, N. Ntarmos, P. Triantafillou, D. Makreshanski, exo: Decentralized autonomous scalable social networking., in: CIDR, 2011, pp. 85–95.

[45] M. Durr, M. Maier, F. Dorfmeister, Vegas–a secure and privacy-preserving peer-to-peer online social network, in: Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom), IEEE, 2012, pp. 868–874.

[46] B. Guidi, T. Amft, A. De Salve, K. Graffi, L. Ricci, Didusonet: A p2p architecture for distributed dunbar-based social networks, Peer-to-Peer Networking and Applications (2015) 1–18.

[47] M. Conti, A. De Salve, B. Guidi, F. Pitto, L. Ricci, Trusted dynamic storage for dunbar-based p2p online social networks., in: OTM Conferences, Springer, 2014, pp. 400–417.

[48] R. I. Dunbar, The social brain: mind, language, and society in evolutionary perspective, Annual Review of Anthropology 32 (1) (2003) 163–181.

[49] 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.

[50] C. Borcea, A. Iamnitchi, P2p systems meet mobile computing: A community-oriented software infrastructure for mobile social applications, in: Self-Adaptive and Self-Organizing Systems Workshops, 2008. SASOW 2008. Second IEEE International Conference on, IEEE, 2008, pp. 242–247.

[51] A. Rowstron, P. Druschel, Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems, in: IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing, Springer, 2001, pp. 329–350.

[52] A. Rowstron, P. Druschel, Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility, in: ACM SIGOPS Operating Systems Review, Vol. 35, ACM, 2001, pp. 188–201.

[53] F. Tegeler, D. Koll, X. Fu, Gemstone: empowering decentralized social networking with high data availability, in: Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE, IEEE, 2011, pp. 1–6.

[54] Y. Afify, Access control in a peer-to-peer social network, Master's Thesis, EPFL, Lausanne, Switzerland.

[55] R. S. Sandhu, P. Samarati, Access control: principle and practice, Communications Magazine, IEEE 32 (9) (1994) 40–48.

[56] Retroshare official documentation, `https://retroshare.readthedocs.io`.

[57] 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.

[58] H. C. Choi, S. R. Kruk, S. Grzonkowski, B. Davis, J. Breslin, Trust models for community aware identity management.

[59] 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.

[60] 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.

[61] B. Carminati, E. Ferrari, A. Perego, Enforcing access control in web-based social networks, ACM Transactions on Information and System Security (TISSEC) 13 (1) (2009) 6.

[62] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, B. Thuraisingham, Semantic web-based social network access control, computers & security 30 (2) (2011) 108–115.

[63] 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.

[64] OASIS, Xacml version 3.0, `http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html` (2013).

[65] 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.

[66] 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.

[67] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: Security and Privacy, 2007. SP'07. IEEE Symposium on, IEEE, 2007, pp. 321–334.

[68] Diaspora foundation, https://wiki.diasporafoundation.org/FAQ_for_users.

[69] 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.

[70] 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.

[71] L.-H. Vu, K. Aberer, S. Buchegger, A. Datta, Enabling secure secret sharing in distributed online social networks, in: Computer Security Applications Conference, 2009. ACSAC'09. Annual, IEEE, 2009, pp. 419–428.

[72] 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.

[73] O. Bodriagov, G. Kreitz, S. Buchegger, Access control in decentralized online social networks: Applying a policy-hiding cryptographic scheme and evaluating its performance, in: Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on, IEEE, 2014, pp. 622–628.

[74] J. Katz, A. Sahai, B. Waters, Predicate encryption supporting disjunctions, polynomial equations, and inner products, Journal of cryptology 26 (2) (2013) 191–224.

[75] A. De Salve, M. Dondio, B. Guidi, L. Ricci, The impact of user's availability on on-line ego networks: a facebook analysis, Computer Communications 73 (2016) 211–218.

[76] 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.

[77] Retroshare team blog, https://retroshareteam.wordpress.com/.

[78] Friendica helpers, https://helpers.pyxis.uberspace.de/help/Groups-and-Privacy.

[79] 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 Transactions on Dependable and Secure Computing PP (99) (2017) 1–1. doi:10.1109/TDSC.2017.2729553.

[80] D.-W. Kwak, J. Kim, A decentralized group key management scheme for the decentralized p2p environment, IEEE communications letters 11 (6) (2007) 555–557.

[81] A. De Salve, R. Di Pietro, P. Mori, L. Ricci, Logical key hierarchy for groups management in distributed online social network, in: Computers and Communication (ISCC), 2016 IEEE Symposium on, IEEE, 2016, pp. 710–717.

[82] A. Castiglione, A. De Santis, B. Masucci, F. Palmieri, A. Castiglione, X. Huang, Cryptographic hierarchical access control for dynamic structures, IEEE Transactions on Information Forensics and Security 11 (10) (2016) 2349–2364.

[83] C. K. Wong, M. Gouda, S. S. Lam, Secure group communications using key graphs, IEEE/ACM transactions on networking 8 (1) (2000) 16–30.