IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 11, Issue 1 (May. - Jun. 2013), PP 20-30
www.iosrjournals.org

OSN: Privacy Preserving Policies

Shahina K[1], Anand Pavithran[2]
(M E S College OfEngineering, India)
(M E S College Of Engineering, India)

Abstract: *Online Social Networks (OSN) have become enormously popular in recent years. OSN enable people to connect with their friends based on sharing details about their personal information, but they present a number of privacy risks to their users. There are di_erent privacy preserving policies or mechanisms to handle the privacy issues. The main such proposed policies are:*
 Safebook: A Privacy-PreservingOnline Social Network Leveraging on Real-Life Trust
 Lockr: Better Privacy for Social Networks
 fl yByNight: Mitigating the Privacy Risks of Social Networking *Persona: An Online Social Network with User-Defined Privacy*
Safebook is a new approach that tackles the security and privacy problemswith a special emphasis on the privacy of users with respect to the application provider in addition to defense against intruders. Lockr improves the privacy of centralized and decentralized online content sharing systems. Third one, aim to mitigate the privacy risk by presenting a new architecture for protecting information published through the social networking website, Facebook, through encryption privacy control. In Persona, users dictate who may access their information. The architecture and implementation of the four mechanisms are different. The different privacy preserving policies have been compared on the basis of three parameters: Trust on OSN provider, Flexible grouping and Type of architecture and found that Persona is relatively better method.
Keywords - *Integrity, Matryoshka, Insider attack, Time to live.*

I. Introduction

The Internets wide adoption has contributed to online social networking sites thriving popularity.Over the past several years, several social networking sites have arisen to facilitate social interactions on the Internet while revolutionizing how online users interact with their friends, coworkers, colleagues, family, and even strangers. Most social networking sites offer the basic features of online interaction, communication, and interest sharing, letting individuals create online profiles thatother users can view.One of the most important issues we must immediately address in this context is the security and privacy of sensitive information provided by the users.

1.1 SECURITY OBJECTIVES IN OSN

In the context of OSNs, we generally identify three main security objectives, privacy, integrity and availability, which come in slightly different flavors than in traditional systems.

1.1.1 Privacy

The protection of the users privacy to be the main objective for SNS. Privacy not only encompasses the protection of personal information, which users publish on their profiles, presumably accessible by their contacts only. Additionally, communication privacy has to be met. Hence, none but directly addressed or explicitly trusted parties may have the possibility to trace which parties are communicating. Privacy calls for the possibility to hide any information about any user, even to the extent of hiding their participation in the OSN in the first place. Moreover privacy has to be met by default; that is, all information on all users and their actions has to be hidden from any other party internal or external to the system, unless explicitly disclosed by the users themselves.

1.1.2 Integrity

As part of integrity, the users identity and data must be protected against unauthorized modification and tampering. In addition to conventional modification detection and message authentication, integrity in the context of OSNs has to be extended: parties in an OSN are not arbitrary devices, but real, unambiguously identifiable persons. In consequence, the authentication has to ensure the existence of real persons behind registered OSN members.

1.1.3 Availability

The data published by users has to be continuously available in OSNs. Availability of user profiles is consequently required as a basic feature, even though considering recreational use, the availability of somecontent may not seem a stringent requirement. In OSNs, this availability specifically has to include robustness against censorship, and the seizure or hijacking of names and other key words. Apart from availability of data access, availability has to be ensured along with message exchange among members.

1.2 SECURITY ANALYSIS OF OSN

First of all, there is a model for social network service, to get an overview on the aim and possible implementation schemes of SNS. SNS can be divided into three different levels (Fig. 1.1):

• A social network(SN) level: The digital representation of members and their
Relationships
• An application services (AS) level: The application infrastructure, managed by the SNS provider
• A communication and transport(CT) level: Communication and transport services as provided by the network

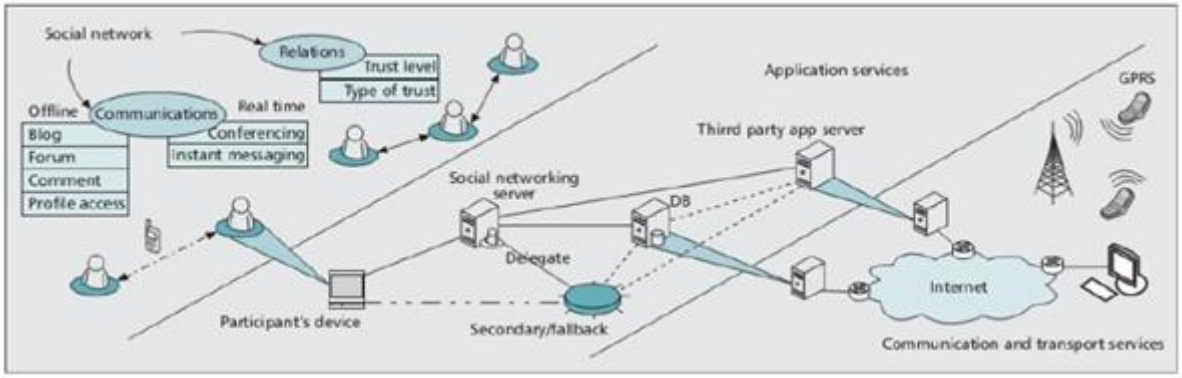

Figure 1.1: OSN levels: three architectural layers of SNS

The SN level provides each member with a set of functions corresponding to social interactions in the real life, like finding friends, accessing profiles, commenting, and the like. To implement these functions, the SN level relies on the AS level. This second level includes the infrastructure managed by the SNS provider, together with basic services to create the SN service, such as web access, storage, and communication. Data storage and retrieval, indexing of content, management of access permissions to data, and node join or leave are implemented in a centralized or decentralized, distributed fashion on the AS level. Other than these inside attackers that primarily seem to be legitimate participants in the system but act in a malicious way in some cases, there may be external attackers, orintruders. An intruder can perpetrate attacks at one or more of the SNS levels.

II. Literature Survey

2.1 PRIVACY PRESERVING POLICIES

There are different privacy preserving policies for OSNs. The four such recently proposed policies are:

• Safebook: A Privacy-PreservingOnline Social Network Leveraging on Real Life Trust
• Lockr: Better Privacy for Social Networks
• flyByNight: Mitigating the Privacy Risks of Social Networking • Persona: An Online Social Network with User-Defined Privacy

2.1.1 Safebook

The privacy of users data is at risk due to the central storage and management and hence threatened by potentially malicious service providers or unintended access following short sighted publication, security breaches, or plain misconfiguration of the OSN. It inherently cannot be ensured with centralized serve based architectures on which all existing OSNs rely. Peer-to-peer architectures seem to offer a suitable alternative to the centralized approach as the basis for a decentralized OSN, avoiding the all knowing service provider. As a major drawback, P2P systems suffer from a lack of a priori trust, thus creating the need for cooperation incentives. Safebook is a decentralized OSN based on a P2P architecture where by basic security and privacy problems as well as the lack of a priori trust and incentives are addressed by leveraging on real life trust between users, such that services like data storage or profile data routing are performed by peers who trust one another in the social network.

2.1.1.1 Safebook: Security Based On Real-life Trust

Safebook consists of a three-tier architecture with a direct mapping of layers to the OSN levels depicted in Fig. 2.1 as follows:

• The user-centered social network layer implementing the SN level of the OSN
• The P2P substrate implementing the AS services
• The Internet, representing the CT level

Each party in Safebook is thus represented by a node that is viewed as a host node in the Internet, a peer node in the P2P overlay, and a member in the SN layer. The nodes in Safebook form two types of overlays:

• A set of matryoshkas, concentric structures in the SN layer providing data storage and communication privacy created around each node
• A P2P substrate providing lookup services

In addition to these nodes, Safebook also features a trusted identification service(TIS), providing each node unambiguous identifiers: the node identifier for the SN level and a pseudonym.

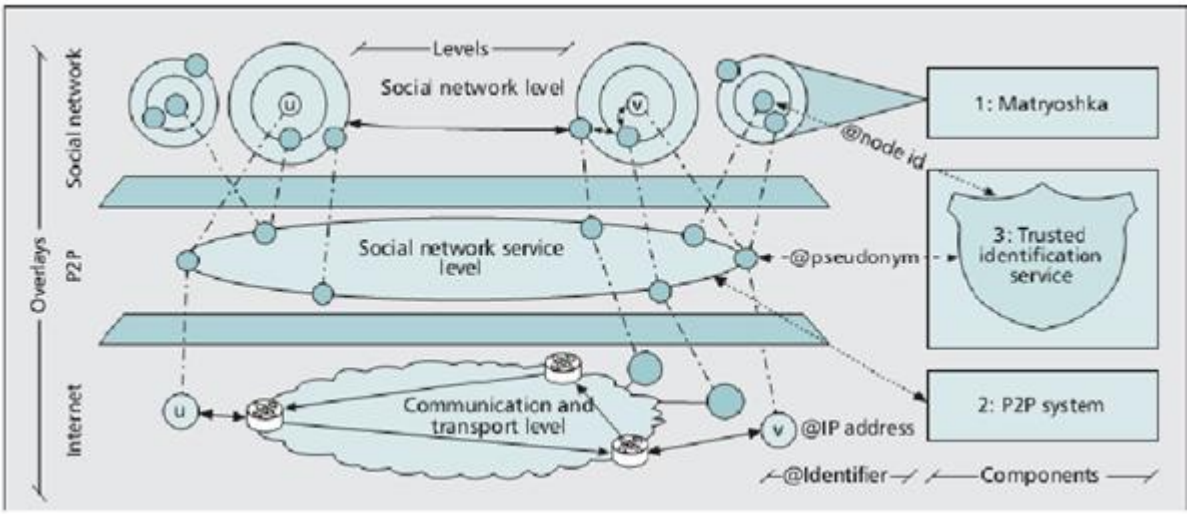

Figure 2.1: Safebook overlays and main components

Matryoshka: Matryoshkas are concentric rings of nodes built around each members node in order to provide trusted data storage, profile data retrieval, and communication obfuscation through indirection. Each matryoshka thus protects the node in its center, the core, which on the SN layer is addressed by its node identifier. The nodes in the matryoshka are connected through radial paths on which messages can be relayed recursively from the outermost shell to the core and vice versa. The innermost and outermost shells of a matryoshka have a specific role: the innermost shell is composed of direct contacts of the core, and each of them stores the cores data in an encrypted form. Hence, they are called the mirrors. Every node in the outermost shell acts as a gateway for all data requests addressed to the core, and is thus called an entry point.

P2P system: In order to provide a location service to find entry points for a users matryoshka, the nodes create a P2P substrate. Unlike the path across a matryoshka, the communication through the P2P layer does not rely on trusted links TIS: The TIS ensures that each Safebook user gets at most one unique identifier in each category of identifiers. Based on an out-of-band identification procedure, the TIS grants each user a unique pair of a node identifier and a pseudonym, computed as the result of a keyed hash function on the set of properties that uniquely identify a party in real life.

2.1.1.2 Operations

Safebook implements different OSN operations:

• Account creation • Data publication
• Data retrieval
• Contact request and acceptance
• Message management

Account Creation: In order to join Safebook, a new member V has to be invited by one of its real life friends A that must already be a registered user. Vs account is then created in the two steps of identity creation and matryoshka creation.

Identity creation: After As invitation, V provides the TIS with its identity property set name together with a proof of owning it. The TIS then computes the node identifier of V and its pseudonym by applying two different keyed hash functions with two different unknown master keys to name. It becomes evident that even if a valid member V repeats the account creation operation multiple times, it will always receive the same

pseudonym and node identifier, since they are a function of Vs identity itself.

Matryoshka creation: V has only A as a contact to start with, so it sends A request for path creation containing the distributed hash table (DHT) lookup keys it wants to register, a time to live (ttl), and the number of members to whom A should forward the request, hereafter called the spanfactor. A then selects between its friends a number span of next hops and forwards them this registration message. This process is recursively done until the ttl expires. The receiving node Dregisters the lookup key in the P2P system together with its reference Id.

Data Publication: The data managed in SNS can be generalized to:

• Profile information
• Contact relations
• Messages

The profile information is the part of the data each user intends to publish. Contact relations represent a members real life relations and can be seen as the friend list of the use. Safebook each user associates a particular trust level to each of its contacts. This level is used to select closely related contacts that primarily will store the published data. Finally, personal messages or comments on profiles can be exchanged between members. In case of comments, the receiver has the right to publish or discard them. To guarantee privacy, data in Safebook can be private, protected, or public: in the first case the data is not published, in the second case it is published and encrypted, and in the third case it is published without encryption.

Data Retrieval: The lookup of Vs data through member U starts with a recursive query in the P2P system: according to the DHT structure, the node responsible for the lookup key responds with the entrypoint list building Vs outer shell. Consequently, U can request that one of Vs entry points forward the request through Vs matryoshka until a mirror is reached.Vs encrypted data then reaches U through the inverse path (Fig. 2.2). The protocol of Safebook uses recursion to hide the source of requests. Additionally, the addressing and routing, for both P2P lookup and data retrieval using the matryoshkas, are based on the pseudonyms of nodes. Attackers consequently have no means to identify a source of a request for some content, as there is no way to distinguish between generated and forwarded requests.

Contact Request and Acceptance: A member U that wants to add another member V to its contact list sends a contact request message following the same steps as in the data request case. Assuming V accepts U as a new contact, V associates with U a certain trust level and sends it back an opportune key that will enable U to decrypt the selected parts of Vs published encrypted data. Message Management: Offline messaging, such as wall posts, recommendations, and other annotations to a profile, is implemented using the steps of retrieving some members data, decrypting the shared parts, annotating some content, and sending this data back, signed with the key bound to the annotators node identifier and encrypted with the public key bound to the receivers node identifier. Real-time messages, like chats, are forwarded to and handled by the core solely and responded to with an error message if the core is offline.

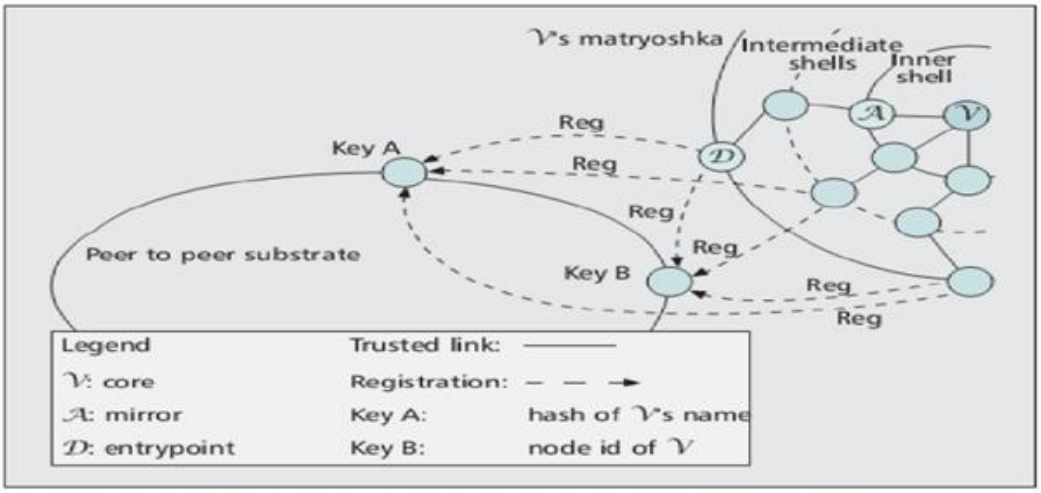

Figure 2.2: Data lookup and retrieval

2.1.2 Lockr

It is a system that offers three key privacy benefits for OSN users.

• Social attestationsto decouple social network-ing information from other OSN functionality:

Lockr users need not provide a full copy of their social network to each online site they use to host their

personal content. Instead, they exchange social attestations: application independent small pieces of digitally signed meta data issued by one person to another that encapsulate social relationships. The recipient of a social attestation can use it to prove the social relationship to any online system.

• The WHPOK protocol to protect social information from disclosure:

Lockr prevents an OSN from re-using the digitally signed social relationships revealed when users access their data by applying to the attestation verification mechanism a variant of zero-knowledge protocols, called awitness hiding proof of knowledge (WHPOK) protocol. This lets users prove the existence of a social rela-tionship without providing a verifiable copy of the attestation that Web sites could store and later reuse.

• Relationship keys to solve social deadlock:

Sharing information in decentralized scenarios, such as peer-to-peer scenarios, give rise to the following problem: two peers would like to exchange content with each other only if they share a specific common friend, yet neither wants to reveal this social relationship to a random peer that does not share this common friend. This creates a form of deadlock, where each peer is waiting on the other one to reveal its relation-ship first. Lockr lets peers encrypt their communication to each other using a key specific to a relationship. In this way, no receiving peer can decrypt a message unless it has the same social relationship as the messages sender.

2.1.2.1 Design Concepts and Components The components that comprise Lockr are:

• Personal Identities and Address Books

In Lockr, a personal identity is simply a public/private key pair. Identities are generated in a decentralized manner, and individuals are in control of how these identities are shared. People can share their public keys with their social networks in the same way they share their names, addresses, and phone numbers. When issuing an attestation to a friend, a user retrieves the friends public key from their local address book and designates it (i.e., the friends identity) as the recipient of the attestation. Thus, people must be careful to record cor-rectly their friends public keys; otherwise, attestations can be issued to nonexistent or even malicious identities.

• Social Attestations

A social attestation is a piece of data that certifies a social relationship. An attestation has six fields: an issuer, a recipient, a social relationship, an expiration date, a relationship key and a digital signature (Figure 2.3 depicts an attestation in XML format). By issuing an attestation, the issuer tells a recipient that they have formed a relationship. Two parties can share more than one attestation since two people can have more than one relationship. Attestations also have an expiration date, and they are signed to prevent anyone from tampering with them.

```
<attestation>
  <issuer>Issuer's public key</issuer>
  <recipient>Recipient's public key</recipient>
  <relationship>
    <type>Relationship type (e.g., family, friend)</type>
    <firstParty>First party's public key</firstParty>
    <secondParty>Second party's public key</secondParty>
  </relationship>
  <expDate>Expiration Date</expDate>
  <relKey>Key specific to encapsulated relationship</relKey>
  <signature>Attestation's signature</signature>
</attestation>
```

Figure 2.3: The XML-based format of an attestation

• Social Access Control Lists (ACLs)

A traditional ACL enumerates the identities of those al-lowed access to certain objects. In contrast, a social ACL does not rely only on identities to specify access control. Instead, they can also allow access to objects based on the social relationship that an individual has with the objects owner. A social ACL contains the owners public key, the public keys of all people who can access the object (as in traditional ACLs), and a social relationship When a user requests access to an object protected by a social ACL, the ACL enforcer first provides the ACL to the requester. The requester then uses it to access an object, the user must either: (1) have their public key listed in the social ACL, or (2) present an attestation issued to them by the owner certifying the relationship listed in the ACL. We also use XML to format social ACLs (fig.2.4).

```
<ACL>
 <owner>Owner's public key</owner>
 <access>
  <user>User's public key</user>
        .......
  <user>User's public key</user>
  <relationship>
    <type>Relationship type (e.g., family, friend)</type>
    <firstParty>First party's public key</firstParty>
   (or <secondParty>Second party's public key</secondParty>)
  </relationship>
  (<and>, <or>, <bracket> additional relationship tags)
 </access>
</ACL>
```

Figure 2.4: The XML-based format of a social ACL

2.1.2.2 Design Goals for Privacy

The main design goals include: (1) putting users in control of their social information by decoupling it from all other functionality of an OSN site, (2) preventing OSN providers from reusing social information revealed by users when requesting access to their friends content, and (3) improving the privacy of peers participating in a P2P online social network

.• Decoupling Social Networking Information from OSNs

Lockr uses social attestations to encapsulate social networking information. Social attestations are OSN independent; they can be issued and exchanged from person to person. There are many convenient ways to issue attestations, such as over e-mail or over a cell phone Bluetooth interface. Lockr does not require the recipient to acknowledge receiving the attestation, although this could be added by a higher level protocol.

• Protecting Social Information

People are reluctant to expose their sensitive social relationships to third party sites. When they have no choice, they may seek assurance that these sites will not abuse the privilege of holding this information by re selling it to others. Lockr is designed to make social information non transferable. Lockr ensures that no one can prove the validity of an attestation other than its issuer and recipient. Thus, when OSN providers learn about an attestation, they cannot transfer this information to others.

• Resolving Social Deadlock

In a decentralized scenario, two peers who do not know each other but share a common friend should be able to verify these social relationships with no loss of privacy. This scenario is challenging because each peer wants to first verify the others social relationship before exchanging content. Lockr solves this problem through the use of relationship keys.

Lockr encrypts an attestation with its relationship key before presenting it to any other party. As its name suggests, the relationship key is specific to a particular relationship; no relationship key can belong to more than one relationship. Its purpose is to protect the confidentiality of the information contained in an attestation during the verification process. The relationship key, shared by all who have that same relationship with the issuer, must also be shared with any entity that needs to enforce a social ACL, such as a third party Web site that hosts the content. This ensures that only people who have a copy of this relationship key can decrypt the attestation.

Lockrs attestation verification protocols make use of the WHPOK protocol and relationship keys to offer its privacy properties. Lockr can perform two kinds of attestation verification depending on the usage scenario. In the first scenario, an OSN site verifies the attestation. In this case, the verification process is one way: the person seeking access uses the WHPOK protocol to verify the attestation. The second is a peer-to-peer scenario. In this case, the verification process is two-way: both peers wanting to exchange content with each other use the WHPOK protocol to verify the attestation.

2.1.3 flyByNight

flyByNight aim to mitigate the privacy risks in OSNs by creating a new architecture for protecting information published through the social networking website, Facebook, through encryption. Our architecture makes a trade off between security and usability in the interests of minimally affecting users' workflow and maintaining universal accessibility.

2.1.3.1 flyByNight Architecture

This application has the following goals:

• Protect personal information transmitted to Facebook by means of encryption
• Ensure that Facebook servers never store cleartext data or private key material and that clear text data never appear on the Internet
• Support one-to-one and one-to-many communication
• Maintain universal accessibility and ease of use of Facebook
• Allow Facebook to manage social network friend relationships
• Use the Facebook interface for key management, storing as much key in formation on the server as possible

The last two goals represent a compromise between usability, security, and privacy. A fully private solution would not reveal even friendship relationships toFacebook; however, this would require abandoning the Facebook architecture, along with its large existing network of friends and useful search features. Similarly, using the Facebook interface for key management introduces some risks, but this design allows users to easily access and use the application from any web browser, requiring only the additional burden on the user of having to remember a single extra password.

The architecture is shown in Figure 2.5. When a user first interacts with the application, he generates a public/private key pair and provides a password. The key generation and other cryptographic operations are performed in client side JavaScript. The password is used to encrypt the private key. The encrypted private key is then transmitted to the flyByNight application server via Facebook servers and stored in a key database on the flyByNight server.

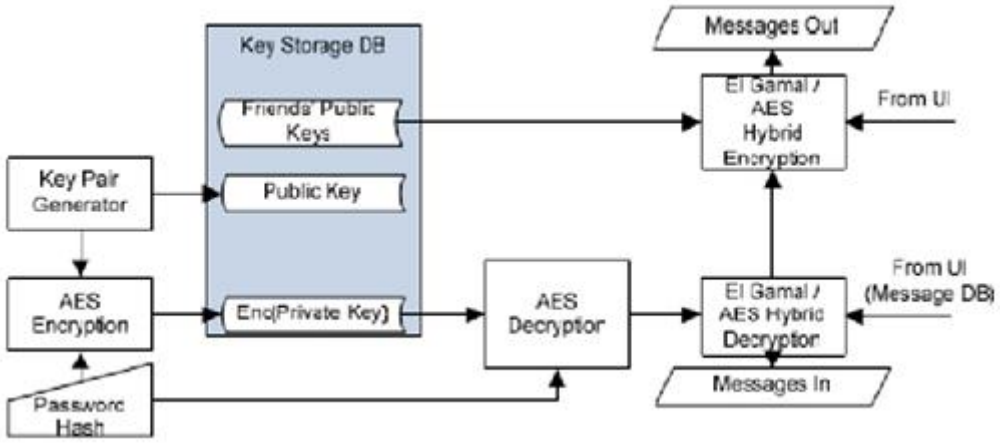

Figure 2.5 flyByNight architecture

When a user wishes to send a private message or make an update to profile or status information, he first enters the message text into the application. The application always has access to a complete list of the users friends who have also installed the application and their public keys, and from this list the user selects which friends to whom he would like the message to be delivered. The client side Java Script encrypts the message with the necessary public keys and tags the encrypted versions with the ID numbers of their intended recipients before sending them via Facebook to a message database on the flyByNight server, where they reside. The existence of these messages is public, but their contents are encrypted.

To read messages, the user queriesthe message database for a list of all encrypted messages sent to him and receives a list of message handles. When a user requests a specific message, the cipher text of the message is delivered. The user decrypts his own private key by supplying his password, and he decrypts the message using his private key. This model can easily be adapted to handle profiles and status messages by allocating in the message database for each user a field to hold his profile text, a field to hold his status message, and so on, with the contents of that field being replaced with a new message containing the users new profile or status every time he makes an update.

2.1.3.2 Implementation

A fully functional prototype implementation of the flyByNight architecture as a Facebook application. The application is accessible through Facebook under the URL http://apps.facebook.com/flybynight. A screenshot of the application in action is shown in Figure 2.6.

Cryptographic Tools:

Implementation is based upon open source JavaScript implementations ofAES and RSA . The first challenge faced was the limitations imposed on JavaScript by the Facebook platform. Facebook rewrites the JavaScript supplied by the application developer to reduce the possibilities of cross site attacks.Variables are

renamed and certain functions are disabled. In addition, Face book imposes a limit on the total size of the JavaScript code that an application is able to use.

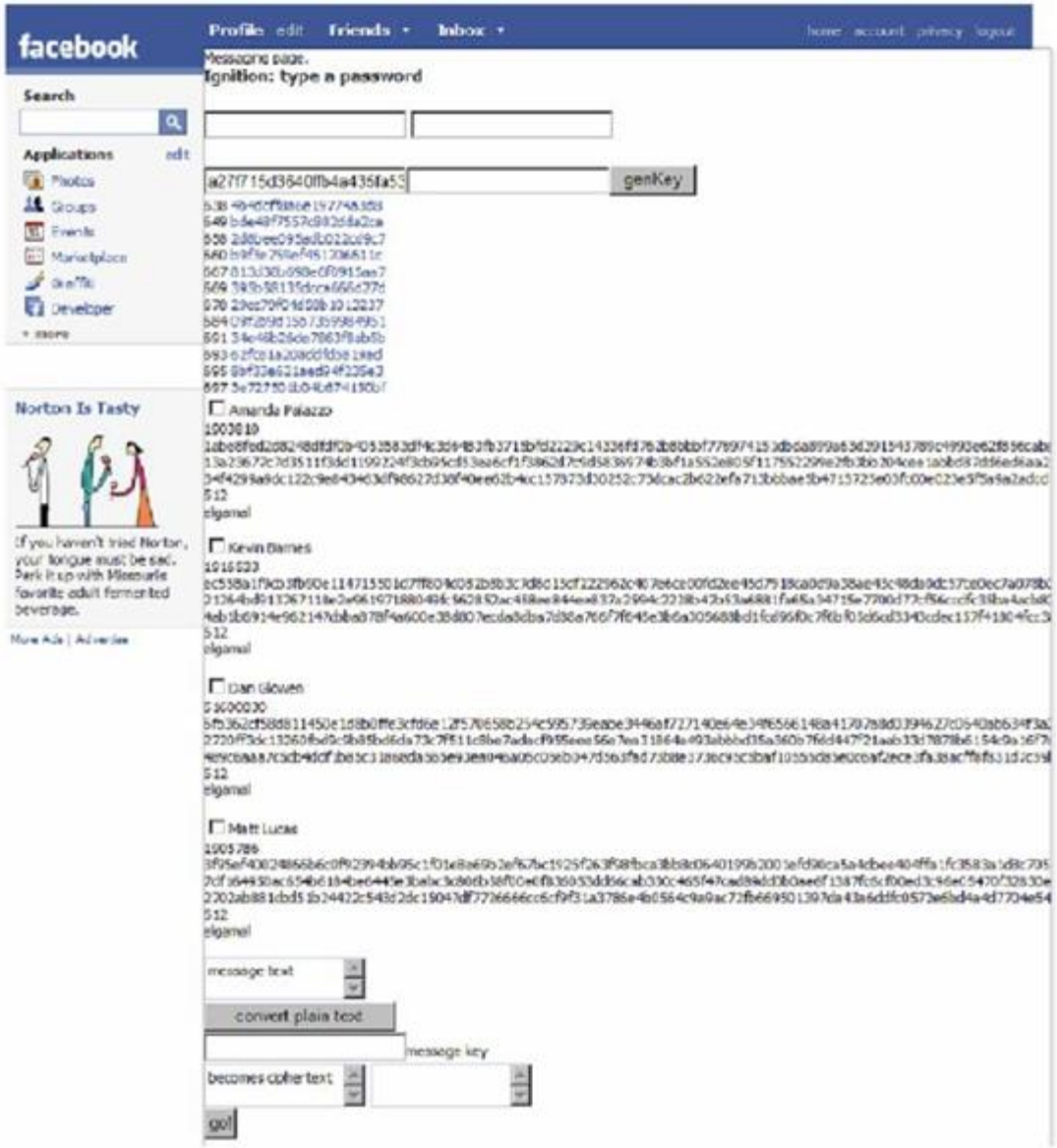

Figure 2.6: yByNight Facebook Application

One-to-Many Communication:

In Facebook, a user typically makes a single update to a profile, status message, or wall that mustimmediately become visible to all his friends at once. To provide a private way to perform such actions, flyByNight supports a one-to-many operation that encrypts a single message for a group of friends. This operation cannot be handled by simple iteration because users commonly have a hundred friends or more, and so a simple implementation of one-to-many encryption would therefore be a hundred or more times slower than a single encryption would, since the message would be individually encrypted for each friend.

User Interface Concerns:

The most important requirement is that the implementation had to be universally accessible as a Web application, just as Facebook is, without any technical knowledge required of the user or binary code. It was imperative that flyByNight maintain the simplicity and platform-independence afforded by Facebook's web interface.

Images:

One goal that was unable to achieve with the flyByNight implementation is encryption of photographs. Photos on Facebook have great potential to be compromising to the individuals pictured in them, and providing them

with the same protection as status updates or one-to-one communication would clearly be desirable. Unfortunately, the JavaScript architecture makes doing so quite difficult. In particular, JavaScript is unable to read local files from a disk. Files must be uploaded to a server first before they can appear in a JavaScript buffer. Such an upload over the Internet in the clear would obviously defeat the purpose of encryption. A helper application to upload photos could be used, but then that would restrict the accessibility of the social network, since users would not be able to upload photos from dumb clients such as mobile phones.

2.1.4 PERSONA

Persona is an OSN where users dictate who may access their information. Persona hides user data with attribute- based encryption (ABE), allowing users to apply fine grained policies over who may view their data. Persona provides an effective means of creating applications in which users, not the OSN, define policy over access to private data.

Persona achieves privacy by encrypting private content and prevents misuse of a user's applications through authentication. Persona allows users to store private data persistently with intermediaries, but does not require that users trust those intermediaries to keep private data secret. Modern web browsers can support the cryptographic operations needed to automatically encrypt and decrypt private data in Persona with plugins that intercept web pages to replace encrypted contents. Lastly, Persona divides the OSN entities into two categories: users, who generate the content in the OSN, and applications, which provide services to users and manipulate the OSN content.

Group Key Management:

Persona users define groups and users generate and use keys corresponding to groups. Keys guard access to two types of objects in Persona: user data and abstract resources. In Persona, all users store their data encrypted for groups that they define. Any user that can name a piece of data may retrieve it, but they can only read it if they belong to the group for which the data was encrypted. Abstract resources represent non data objects, for example, a user's storage space or a Facebook Wall.We use the notation shown in Figure 2.7 in the algorithm listings.

2.1.4.1 Operations

Persona operations allow users to manage group membership and mandate access to resources. The operations combine ABE and traditional cryptography, allowing individuals to be securely added to groups de_ned using ABE and allowing group members authenticated access to abstract resources.

Define Relationship:

Users invoke the DefineRelationship function to add individuals to a group.The user generates an appropriate attribute secret key using the ABEKeyGen function, encrypts this key using the target user's public key, and stores the encrypted key on her storage service. The target user can retrieve this encrypted key.(Fig 2.8)

DefineTransitiveRelationship:

The DefineTransitiveRelationship function allows a user Alice to define groups based on a group defned by another user, Bob. Alice creates a new attribute to describe the new group bob-friend and generates an ASK bob-friend with that attribute. Alice encrypts ASK bob-friend with the access structure using Bob's attribute public key and stores the ciphertext on her storage service. (Algorithm2 in Fig 2.9) Users with the attribute may retrieve and decrypt this key and use it to view the content encrypted within Alice's ABE domain. Alice may include a traditional keypair, used for authentication to ACLs, in the cipher text C

Term	Definition
$u.SS$	u's storage service location
$u.K$	Key K created by u
(TPK, TSK)	PKC public/secret keypair
$(APK, AMSK)$	ABE public/master secret keypair
ASK	ABE user secret key
AS	Access structure
TKeyGen()	Generate RSA keypair
TEncrypt(K, m)	RSA encrypt m with key K
TDecrypt(K, c)	RSA decrypt ciphertext c
TSign(K, m)	RSA sign m with key K
ABESetup	Generate an attribute public key and master secret key
ABEKeyGen$(K, attrs)$	Generate an attribute secret key with attributes $attrs$
ABEEncrypt(K, m, AS)	ABE encrypt m with key K and access structure AS
ABEDecrypt(SK, PK, c)	ABE decrypt ciphertext c with secret key SK

Figure 2.7: Notations

Algorithm 1 : DefineRelationship(u1, attrs, u2)

u1: A ← $ABEKeyGen(u1.AMSK, attrs)$

u1: u1.SS.put(H'(u2.TPK),C)

...

u2: C← $u1.SS.get(H'(u2.TPK))$

Figure 2.8: Algorithm 1

Algorithm 2: DefneTransitiveRelationship(u1, APK , access structure AS, attrs)

u1: A← $ABEKeyGen(u1.APK, attrs)$

u1: C← $ABEEncrypt(APK, A, AS)$

u1: u1.SS.put(H'(AS, APK), C)

Figure 2.9: Algorithm 2

AssignRightsToIdentity:
Resource owners use AssignRightsToIdentity to provide other users specifirights to named resources. An example of such a right would be the ability to storedata on another user's storage service.(Fig 2.10)

Algorithm 3: AssignRightsToIdentity(u1, rights, TPK, resource r, owner o)

u1: o.chACL(r, TPK , rights)

Figure 2.10: Algorithm 3

To assign rights, the user instructs the resource's home to add a (public key,set of rights) pair to the resource's ACL. If the public key was already in the ACL, then the rights are changed to those specifed in the new rights set (Algorithm 3).
AssignRightsToGroup:
The AssignRightsToGroup function allows a user Alice to provide resource access to a group G rather than to an individual.The group is speci_ed using attributes de_ned in Alice's ABE domain. (Algorithm 4, Fig 2.11).
First, Alice creates a new (TPK ,TSK) pair speci_cally for G. Alice ABE-encrypts this keypair with an access structure that identi_es members of G. Alice stores the resulting ciphertext on her storage service. This pair of PKC keys becomes the group identity and Alice can assign rights according to AssignRightsToIdentity.

Algorithm 4: AssignRightsToGroup(u1, rights, access structure AS, resource r, owner o)

u1: (TPK , TSK)← $TKeyGen()$

u1: C← $ABEEncrypt(u1.APK, (TPK, TSK), AS)$

u1: u1.SS.put(H'(AS, APK), C))

u1: AssignRightsToIdentity(u1, rights, TPK , r, o)

Figure 2.11: Algorithm 4

2.1.4.2 Revocation of Group Membership

Removing a group member requires rekeying: all remaining group members must be given a new key. Data encrypted with the old key remains visible to the revoked member. The nominal overhead is linear in the number of group members but can be reduced . An ABE message can be encrypted with an access structure that specifies an inequality and the message can be decrypted only if a user possesses a key that satisfies the access structure.

2.1.4.3 Publishing and Retrieving Data

Private user data in Persona is always encrypted with a symmetric key. The symmetric key is encrypted with an ABE key corresponding to the group that is allowed to read this data.This two phase encryption allows data to be encrypted to groups; reuse of the symmetric key allows Persona to minimize expensive ABE operations. Users put (encrypted) data onto their storage service and use applications to publish references to their data. Data references have the specific format. The tag and storage service specify how to retrieve the encrypted data item, and the key-tag and key-store specify how to obtain a decryption key.

III. Performance Comparison

Privacy concerns in OSN have been repeatedly raised in the last few years. There are a few architecture that provide the ability for users to protect their privacy. TABLE 3.1 summarizes the comparison of important privacy preserving mechanisms proposed for OSNs.

Safebook is completely trusted on OSN provider and adopts decentralized architecture. Flexible grouping is available here. Lockr trust OSN provider and a third party server. It is centralized and exible grouping is also possible.The third one, flyByNight relies on trustworthy of both yByNight servers and OSN. The user can only communicate with one group at any given time and adopts a centralized architecture. Persona provides an effctive means of creating applications in which users, not the OSN, define policy over access to private data.

It uses decentralized architecture and flexible grouping is also possible here. So it is the better method among these policies.

Table 3.1: Comparison of privacy approaches proposed for OSNs

Parameter	Safebook	Lockr	flyBytNight	Persona
OSN Provider	Trusted	Trusted	Semi-trusted	Un-trusted
Architecure	Decentralized	Centralized	Centralized	Decentralized
Flexible grouping	Possible	Possible	Impossible	Possible

IV. Conclusion

There are many benefits in joining the social networks, but online users require some restricting mechanism on access to their personal data. There are a several architectures that provide the ability for users to protect their privacy. Four types of recently proposed privacy preserving policies on OSNs are: (1). Safebook; (2). Lockr; (3). yByNight; (4). Persona. These are compared using different performance parameters such as trust on OSN provider, type of architecture, and possibility of flexible grouping. It is concluded that Persona is relatively better method among these policies based on the specified parameters.

References

[1] L.A.Cutillo,R.molva, and T.Strufe Safebook:A privacy-preserving online social network leveraging on real-life trust, *Communications Magazine, IEEE,vol.47, no.12, pp.94-101*, Dec. 2010

[2] Amin Tootoonchian, Stefan Saroiu,and Yashar Ganjali, Lockr: better privacy for social networks, *CoNEXT '09.ACM* Dec.2009

[3] Lucas Matthew M, Nikita Borisov, flyByNight: Mitigating the Privacy Risks of Social Networking, *WPES Proceeding ACM* 2009.

[4] Randy Baden, Adam Bender, and Neil Spring, Persona: an online social network with user-defined privacy, *SIGCOMM '09 Proceeding,Vol.39, pp.135-146, ACM* 2009