Privacy in Social Networks

Bachelor thesis in the course of studies
Media Informatics

Submitted by Yaron Strauch at Stuttgart Media University on July 17th, 2016, to obtain the academic degree of Bachelor of Science

First examinator: Prof. Walter Kriha
Second examinator: Sebastian Luxem

Abstract

Online Social Networks (OSNs) are heavily used today and despite of all privacy concerns found a way into our daily life. After showing how heavy data collection is a violation of the user's privacy, this thesis establishes mandatory and optional requirements for a Privacy orientated Online Social Network (POSN). It evaluates twelve existing POSNs in general and in regard to those requirements. The paper will find that none of these POSNs are able to fulfill the requirements and therefore proposes features and patterns as a reference architecture.

Contents

1 Motivation

Online Social Networks (OSNs), as defined by Boyd and Ellison [1], allow users to create profiles, add users as friends[1], visit profiles of other users, and send messages either privately or publicly. As this definition matches a variety of OSNs, the author decided to mainly concentrate on the OSN with the most active user count, which at this time[2] is Facebook [3].

So how does Facebook deal with data and how does this behavior affect the privacy of its users?

Shortly, Facebook collects all data of its users. Data can be shared willingly or recorded via analytics software on Facebook itself, on connected devices, or on any external web site that uses Facebook services like a comment box or a share button [4, *What kind of information do we collect?*]. They don't only collect data from registered users, but from any third party users are interacting with. That means even data from unregistered users is collected and combined, resulting in much greater information value than users shared willingly.

The resulting data is not only used for Facebook's core services such as messaging and communication feeds, but also for personalized advertising of Facebook and all of its partners [4, *How is this information shared?*]. Facebook claims not to share data that "personally identifies you [...] like name or email address that can by itself be used to contact you or identifies who you are". This phrasing allows Facebook to share even very sensible personal information like birth date, work places or GPS data with other companies.

One could argue that users entered their data willingly into the system and therefore decided to share this data with Facebook. But data that was recorded on Facebook applications or even third party websites was not entered willingly. Additionally one cannot regulate what information other users share about oneself.

But more importantly one cannot regulate what new data is inferred. There are algorithms that allow Facebook or third parties to infer both identifying and additional information that was neither recorded via analysis nor explicitly entered by the user. A development team was able to infer "with as little as 20% of the users providing attributes [...] the attributes for the remaining users with over 80% accuracy" [5]. Michel Kosinski et al. were able to predict very private information with a high accuracy such as sexual orientation, religion, political views, intelligence, or happiness solely based on Facebook likes [6]. Other attacks use the Facebook group memberships of a victim [7] or mutual

[1] This paper uses the term *friend* in the same way as current social networks do: Besides of what *friend* commonly refers to, they also categorize other contacts one interacts with such as family members or co-workers as *friend*.

[2] It is unclear if Facebook will keep this position since a (heavily criticized) study from 2014 predicted a massive decrease in Facebook usage [2]. But because today one and a half billion users use Facebook regularly [3] this paper will use Facebook as a reference.

friends [8] to reveal one's identity or personal information.

So Facebook and other companies are able to generate new data that a user never shared willingly and that may be very sensitive and private. But since the data doesn't personally identify the user, it is shared with every associated company.

Since people cannot control what data their contacts share about themselves, which information may be inferred, and what data is recorded, the system becomes untransparent and uncontrollable. Users may change privacy settings in order to hide information from other users. But this seems more like the illusion of privacy. A study showed that "Most users do not seem to realize that restricting access to their data does not sufficiently address the risks resulting from the amount, quality and persistence of the data they provide" [9]. In other words, changing privacy settings within Facebook does not affect sharing data with Facebook or any associated company [4].

Most users don't seem to be aware of the value of their private information. A study that evaluated a questionnaire on more than one thousand people tried to figure out how much money people value their Facebook data. It showed that nearly half of them valued their data as 0 EUR [10]. All evaluated participants chose to spend between 0 EUR and 150 EUR, where the average would spend between 5 EUR and 15 EUR. This study shows that people either don't understand or ignore the fact that Facebook generates billions from the data they provide [11]. Plus they don't know what possible value inferred data may have.

Past events already showed that even unremarkable data may have a great impact on one's life. In 2009 an insurance company stopped paying a depressed woman after seeing a photo where she seems to have fun [12].

One may argue that data collection is part of Facebook's effort to increase user experience. And some might find it acceptable that their data is collected and processed through face recognition, location tracking, and user behavior recording. But there are services and algorithms completely unrelated to social networks that definitely don't increase user experience within an OSN. For example, Facebook registered a patent for an algorithm calculating credit ratings using the social graph [13]. That means if your friends are considered uncreditworthy, your own financial credibility is ranked down. Facebook may even calculate credibility through profile, image, and location analysis. If users do not want to be ranked based on their social profile, they cannot opt out or object without deleting their whole account.

If personal data was encrypted in such a way that Facebook was not able to read it, all those leaks and attacks on the user's privacy would be impossible. The users, unaware of the real value of their data, could be protected against privacy violations by a fully encrypted OSN.

Generally privacy and encryption are closely related. In a report to the Human Rights Council, David Kaye pleads for the use of encryption and anonymity tools [14]. He states that it was "recognized that privacy is a gateway to the enjoyment of [...] the freedom of opinion and expression" and derives that "Surveillance systems, both targeted and

mass, may undermine the right to form an opinion, as the fear of unwilling disclosure of online activity, such as search and browsing, likely deters individuals from accessing information". For Kayle, secure communication "may be the only way in which many can explore basic aspects of identity, such as one's gender, religion, ethnicity, national origin or sexuality".

Since 2015 almost 20% of Internet usage belonged to Facebook [15] and social networks are essential tools to explore what Kayle calls "basic aspects of identity", the need for a fully encrypted social network should be obvious and an important goal to achieve.

2 Research questions

This thesis will discuss the following research questions (RQs):

1. What are mandatory requirements for a Privacy orientated Online Social Network (POSN)?
2. What are additional criteria that are not necessary, but nice to have?
3. a) Is there already a POSN that fulfills the given requirements?

   b) If not: How can one approach a POSN with the given requirements?

Method

Requirements are considered essential if they are needed to guarantee privacy or data integrity[3]. Because a POSN needs to have a chance to compete with current OSNs like Facebook, requirements are also considered essential if they are needed to oppose Facebook.

This list of requirements is a proposition and not required to be comprehensive. It is merely a starting point for RQ3 in order to provide a base to evaluate existing software. It is possible that the list will change after the evaluation phase.

[3] In order to simplify readability, the term *data integrity* is used to refer to *authenticity* as well

3 Requirements for POSNs

3.1 Mandatory requirements for POSNs

End to end encryption

The core of an OSN is messaging. Messages are either public (announcements of official pages such as companies), one to one (private, direct messages), or one to many (status updates, profile comments). Public messages are not needed to be encrypted.

If a message is not considered public, there is always a fixed set of recipients. In a private message there are two people communicating and no one else should be able to decrypt it. If someone posts a status update, every contact of the sender is a recipient and therefore only contacts should be able to decrypt it.

End to end encryption implies that only the recipient is able to decrypt the message. The decryption needs to take place on the client, not on the server. Keys must not be stored anywhere but on client side. The server must not be able to decrypt data or see any plain text.

Data integrity

Every message, whether sent publicly or privately, needs to be signed. The recipient needs to check the signature in order to ensure that no man in the middle attacks took place. This may be achieved by a CA system, but since a social network is built on people trusting each other in real life, a decentralized trust system may be a better choice. For users that don't know much about encryption, a simple system to state possible security breaches in an understandable way (like a traffic light system) must be integrated. Trusting or untrusting people need to be easy. A proposition would be to present a list of friends that already trust the new party, allowing the user to judge based on already established trust. Another option may be a QR validation code for trusting new parties, as the encrypted instant messenger Threema is currently using.

Usability

To compete with Facebook, a POSN needs to be as intuitive and as easy as familiar tasks like using Facebook or installing flash player. If a program needs the users to use a command shell, install multiple software dependencies, research bugs, or understand what asymmetrical cryptography means, the system would be far behind Facebook and couldn't compete. Instead, good usability patterns are needed. Software needs to be self explanatory and have sufficient user documentation.

Mobile Friendly

Since mobile devices are as common as desktop devices and faster growing [16], a mobile app or at least a mobile optimized, well functioning web client is needed. Whether the mobile device acts as a server (in a peer to peer network) or if it depends on a machine to synchronize with needs to be discussed using a concrete software. If it is a native application, it should be be installed like any other ordinary application.

Consistency

What consistency models are to be expected?

Twitter uses two models, it can take both eventual and strong reads [17], where strong reads follow the consistency model of *sequential ordering of per-object updates*. Sequential ordering of per-object updates means that as if two writes occur after each other, first setting X=1 followed by X=2, a reader may read (1,1), (1,2) or (2,2), but never (2,1).

Facebook uses a consistency model called "Existential Consistency" [18]. They claim "Overall Facebook's design provides per-object sequential consistency and read-after-write consistency within a cache, and eventual consistency across caches [...] we expect most of them [user's sessions] to receive per-object sequential and read-after-write consistency". While this guarantees a sequential consistency for interactions between people that are geographically close to each other and therefore on the same cache, it also means that people farer away experience eventual consistency.

Both networks claim that eventual consistency leads to anomalies not suitable for social networks [17, 18]. Therefore it is mandatory to have a stronger consistency model than eventual consistency, at least for people geographically close to each other.

Availability

Availability is one of the key requirements for websites in general. A social network and all of its components such as news feed or chat should always be available to the user. It is not acceptable if some services were completely disabled due to offline hosts.

3.2 Optional requirements for POSNs

Open Source

The debate if Open Source increases security has not ended yet. There are multiple papers analyzing if Open Source increases security and some come to the conclusion that it is not dependent on whether code was published or not, but rather on the number of reviewers that looked into the code, how skilled they are, and how well structured the code is [19, 20, 21]. If those reviewers look into opened or closed source code seems not to matter. Therefore this thesis will see Open Source as an optional requirement.

Distribution

Facebook's expenses in 2011 were 860 Million USD [22]. When the system is not able to generate money using the user's data, how can servers and bandwidth be financed? Some of the discussed OSNs are distributed in a peer to peer manner, allowing to use every user's machine and reduce server cost. Other projects are distributed onto different servers, allowing do distribute resources and therefore cost among multiple owners.

Another advantage of a fully distributed network would be data access since a hacker wasn't able to monitor all the traffic (meta-)data. In a paper of Amre Shakimov et al., three categories of decentralized OSNs are described [23]. The first one is storing the personal data on a highly available cloud storage. The second approach uses the user's machine and the machines of trusted friends of the user, being cheaper because no cloud service is required, but not as available as the first approach. The third approach is a hybrid system between these two.

As long as the data is encrypted, all three models are acceptable in a distributed system, always with the trade off between availability and cost. If data is fully encrypted, a distributed storage is not necessarily needed. This is why this thesis considers distribution as an optional requirement.

There is, however, a problem with heavily distributed peer to peer systems and mobile devices. Today mobile devices have got limited traffic volume, network speed, disk and battery capacity, and computing performance compared to desktop computers. Since all these factors will further increase in the future, it may be that at some day they will be a regular part of peer to peer systems. Further evaluation of this thesis is not part of this paper and may fill a whole other future study.

Obfuscated Social Graph

Whilst an end to end encryption is mandatory, it may not be possible to encrypt every information. For example in a classic peer to peer application there is a master server that knows about which IP belongs to which user id. But hiding the social graph at least partially or obfuscating it would be a great security improvement, since it protects against the mutual friend attack [8].

Optional identity validation

Kayle argues "anonymity may liberate a user to explore and impart ideas and opinions more than she would using her actual identity" [14]. But at the same time users of a social network need a mechanism to verify one's identity. Facebook validates users using their mobile phone number. Such a central authority to validate identities could be counter productive to the idea of minimizing centralism. There are, however, possibilities to verify users in a decentralized manner, such as trust chains. An optional verification system would satisfy both the need for anonymity and the need to verify one's contacts. A decentralized validation system is preferred.

4 Analysis of existing approaches for POSNs

How to approach a POSN? In section 4.1 this paper will examine projects that designed POSNs as a whole, followed by section 4.2 discussing existing frameworks to build POSNs on. Finally in section 4.3 it will analyze projects that enhance privacy within already existing OSNs.

4.1 POSN projects

4.1.1 Diaspora

Features

Diaspora [24] is a heavy distributed OSN with focus on privacy.

Users create their account within a *pod*, meaning a server in a peer to peer network. They can decide if they want to join a pod or if they want to create a pod on their machine.

A pod works very much like an email server. Users register to a pod, receive a unique identifier ending with *@<server address>*, and may communicate with both, users on the same pod and users on other pods. Email servers from different providers only synchronize data when needed, for example when an email is sent between them. In the same way the diaspora network does not synchronize in a mirroring manner, but only partially on demand. Pods synchronize content with each other using a direct HTTPS connection.

Messages are either shared publicly or addressed to *aspects* that allow the user to comprehend what data is visible to others. In the same manner the news feed may be filtered using aspects.

In contrast to Facebook, Diaspora doesn't want its users to identify themselves with a real identity. They may use pseudonyms to use diaspora in an anonymized way.

Evaluation

Diaspora does not encrypt its data in an end to end manner. While most Diaspora servers use HTTPS and therefore ensure only client to server encryption, it is even possible to use it only with HTTP. However, private messages are encrypted between two pods using PGP [25]. Note that this is not an end to end encryption, but an encryption between the two pods. The pods may even store the private messages in plain text. Even though diaspora provides better privacy because data is not saved on a central machine but in a decentralized manner, a pod owner is still able to look into the plain texts.

Registering to a pod and using diaspora is not complicated and as easy as using Facebook. Setting up a pod, however, needs time and administrative skills. Ordinary users don't know about how to set up a database, web server, reverse proxy, and HTTPS certificates, meaning creating a pod is a rather huge obstacle for an ordinary user. This

results in pods with over 10.000 users. If users register without creating a new pod they may pick from a list of pods or let the diaspora system pick a pod for them.

The Social Graph is not shared with users or other pods. Only one or two the required pod administrators and users know who the user shares information with. This works because sharing is asymmetrical, which means there are no friend requests like in Facebook. A user may share content with a second user without acquiring permissions.

The aspect system creates very transparent communication channels and is a great step into an easy to use privacy management. Users are always aware who they are sharing posts with without complicated privacy settings we currently find on Facebook.

Diaspora is open sourced and heavily updated for security fixes [26].

Since a pod owner is able to read every message from its users and there are pods with thousands of active users it should be discussed if diaspora's model solves privacy violations. A pod owner could still eavesdrop on private messages. Of course the potential power of administrators is reduced due to the distributed model, but it is not eliminated. Even users that have their own pod are not safe from privacy violations, because their personal data is synchronized to all the pods their contacts use. If one contact they are interacting with is registered to a malicious[4] pod, their private profile and every message sent to that specific user may be monitored by that pod. Since there is no end to end encryption the users have no chance to protect themselves against such violations. Users are required to trust their pod owner and the owner of every pod owner their contacts are registered on.

Diaspora uses a lose trust chain. Assuming every user trusts their pod owner and every user trusts their contacts, they claim this means every user trusts every pod owner of their contacts. This would be viable if everyone had a private or trustworthy (e.g. family only) pod and protects it against attackers and malicious software. But since a normal user is neither able to create a pod, defend it against attackers, nor understand the impact of who to trust (and maybe just picked a pod randomly), this trust chain does not work.

[4] In this paper the term *malicious* is used not only for evil attackers, but for formerly trustworthy, compromised servers as well

Table 1: Criteria evaluation of Diaspora

Mandatory Requirements	
Criterion	**Outcome**
End to end encryption	No
Data integrity	No
Usability	Usage: Easy Administration: Hard
Mobile friendly	Mobile optimized Web Page. No native application
Consistent	High
Available	Pod needs to be always online
Optional Requirements	
Criterion	**Outcome**
Open Source	Yes
Distributed	Multiple servers
Obfuscated Social Graph	Pod Administrator sees contacts
Identity validation	No

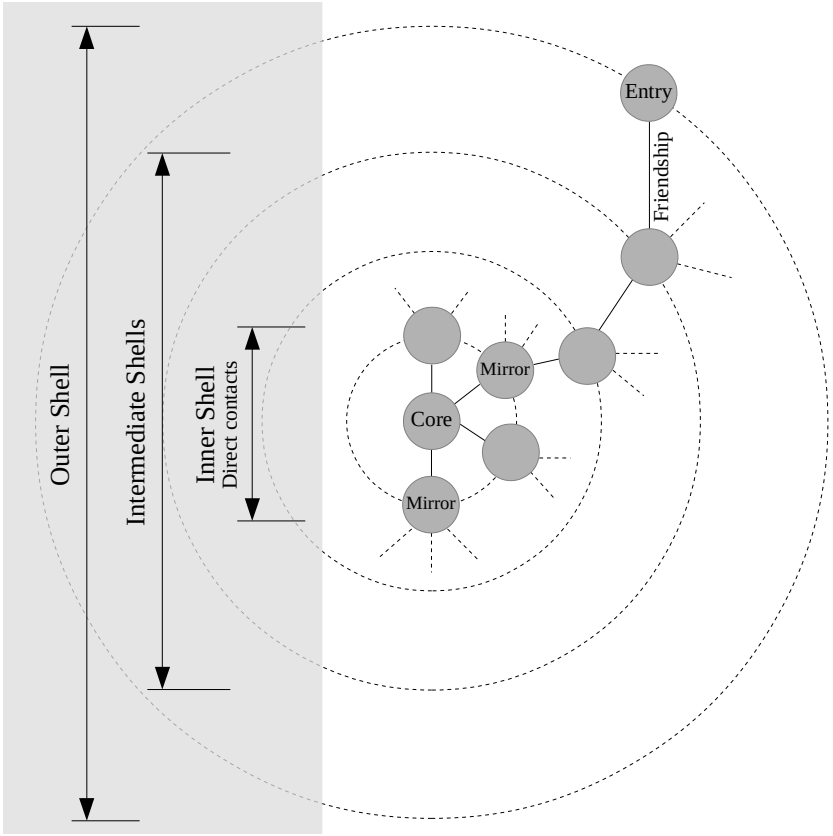
4.1.2 Safebook

Features

At the time this paper was written, safebook was still in development. All information relies on a concept of the development team [27].

Safebook is an encrypted, peer to peer social network. Its key feature is a trusting network based on real life trust, where each user is represented by a node. Such nodes stand not only for a user in the social network, but also for a host node in the Internet and a peer node in a peer to peer network.

Figure 1: A Matryoshka



Data is saved and routed in *matryoshkas* (See Figure 1), a structure to both mirror data and obfuscate routes in the network. A matryoshka consists of a user's node (*core*) and multiple other nodes built at multiple layers (*shells*) around it. All paths between nodes are based on a real life trust relationship. Messages hop along those paths, similar

to IP packets. The innermost shell around the core represents direct contacts of the user and a subset of its nodes replicate the core's data in an encrypted storage. On the outermost shell there are nodes that represent *entry points* to the core and act as gateways to the user.

As long as the user's server is online it will handle all requests itself. If it is offline, data is served by one of the replicas from the innermost shell.

Safebook uses a *trusted identification service* to give nodes unique identifiers. Even though being implemented as a central service, it only receives a hash over personal data to generate these ids and therefore is not a security issue to the network. It cannot decrypt data or even get meta data on information sent between clients. It may be implemented in a decentralized manner as well.

Evaluation

End to end encryption is implemented via asymmetrical encryption. Safebook generates two key pairs on the client side. The first keypair is the pseudonym keypair, used to encrypt and sign routed messages between the nodes of every hop. The other keypair is called node keypair, ensuring an end to end encryption and verification. So both criteria, end to end encryption and data integrity, are matched.

Safebook specifies all communication should be handled by the owner's node if possible. If the node is offline, the node's replicas serve the information. Real life components like the chat function must be served through the owner's node and become deactivated if the owner's node is offline. This means not every function is available all the time and therefore the availability requirement is not matched.

Since safebook is not released yet, it is not possible to analyze usability. A mobile client is not yet implemented and it is not mentioned if there is one in development. It may be difficult to implement a mobile client as long as it is specified that all real life communication must be handled by the user's node.

The authors of safebook did not write anything about the consistency model. As long as there is no prototype or source, one may only speculate on the consistency model. Since safebook relies on a distributed peer to peer system, the author assumes it to be eventual consistent.

Because only close friends are used to mirror data and ordinarily close friends are geographically close to each other, availability might not be given at night since their personal computers don't run.

The matryoshkas fulfill the optional requirement of an encrypted social graph very well. Only close friends can contact each other and entry points for other people are given, but an attacker can never know or learn the whole social graph.

Safebook is a very promising and fascinating concept for POSNs. It eliminates privacy issues by design and focuses on encryption and obfuscation. It is unknown how performant and reliable an implementation will work, how usable it will be or how to

implement a mobile client. Without a prototype or a public proof of concept everything is built on speculation. Since 2010 the authors of safebook didn't publish new articles or released a prototype, it is in question if safebook is still in development or ever released.

Table 2: Criteria evaluation of Safebook

Mandatory Requirements	
Criterion	**Outcome**
End to end encryption	Yes; Trust chains built on real life trust
Data integrity	Yes
Usability	Unknown
Mobile friendly	No
Consistent	Unknown; eventual consistent?
Available	Not when host and close friends are offline
Optional Requirements	
Criterion	**Outcome**
Open Source	No
Distributed	Peer to peer
Obfuscated Social Graph	Yes
Identity validation	Not natively

4.1.3 PeerSoN

Features

PeerSoN [28] is a prototype for an encrypted peer to peer OSN.

As many peer to peer systems, PeerSoN consists of a centrally managed lookup service, currently implemented with open Distributed Hash Table (*openDHT*), and the peer to peer network serving the user data.

The prototype focused on distribution and not on encryption. The developers propose to use a public key infrastructure or a hybrid encryption. Currently every user owns a private key, the key distribution is not discussed.

Each user is identified by a globally unique ID (*GUID*) which is currently implemented as a hash of the user's email address. Each user may be logged in on different machines (*locations*). Those locations and their online state are saved onto the lookup service, allowing other users to request if and where the user is currently logged in. When a user wants to receive updates from another user, a request to the DHT is sent to read the index file of the required user's profile. The index file contains links to all file names that in sum are the user's profile. The DHT returns which machine from the peer to peer network is able to serve the latest version of that index file and which version the file is on. The machine serving the file doesn't need to be the owner of that file.

PeerSoN uses asynchronous messages in order to provide services even when a client is offline. In the prototype this is implemented within the DHT. Saving the messages into the DHT is not considered a security thread as messages are always encrypted using the recipient's public key.

Evaluation

PeerSoN is only a prototype and therefore the developers did not implement access controls. Further they documented problems with openDHT. Asynchronous messages that are saved onto openDHT are capped to a maximum length of 800 characters, they are deleted after a week, and the openDHT may be slowed down when malicious clients perform a DDOS attack. Despite OpenDHT being technically decentralized, they documented that it is provided by a third party and therefore owned in a centralized manner. Due to that logical centralization it hold by a single third party and contradicts a heavily distributed system. The developers think about implementing a specialized DHT specifically for PeerSoN. This should be not a big problem since the DHT only serves meta data such as IP addresses and therefore the implementation may be changed easily to another distributed storage.

The project was built with mobile clients in mind, they designed the direct connections to end immediately after the file transfer finished in order to save data volume. Mobile devices don't need to serve content either and future security measures may include that clients can deny connections during the handshake to prevent the download of big files.

The developers documented problems in view of availability and heavy disk usage due

to the need of heavy data mirroring. PeerSoN is not finished, but it documents how hard it is to build an OSN on top of current peer to peer systems.

Table 3: Criteria evaluation of PeerSoN

Mandatory Requirements	
Criterion	**Outcome**
End to end encryption	Yes; Key distribution is not implemented
Data integrity	Yes
Usability	Unknown
Mobile friendly	Yes
Consistent	Messages are delivered instantly or eventually
Available	Medium to High
Optional Requirements	
Criterion	**Outcome**
Open Source	No
Distributed	Peer to peer
Obfuscated Social Graph	Unknown
Identity validation	Not natively

4.2 Existing Frameworks for POSNs

4.2.1 FOAF

Features

FOAF [29] is a technical architecture to provide POSNs. In contrast to current OSNs that serve HTML/CSS/JS it proposes to build a decentralized and open network that serves raw data. FOAF means *Friend-Of-A-Friend* and describes a file that stores the contacts of a user.

A user is represented by a *Web ID*, which is a URI to point to the user's FOAF file on a server the user trusts. In the FOAF file all contacts are listed with their Web ID. This provides different users to reside on different servers and still be friends. Because the FOAF file also provides URIs for interactions like reading or writing posts, users may even use multiple servers to serve their profile.

Every server may implement access control measurements like openID themselves, making the system highly flexible. Different access control may even lead to features Facebook doesn't provide like collaborative editing or write-only mail in-boxes.

Since FOAF itself is a formatting language it is not sufficient for users, they need a viewer to interpret and display the FOAF files and to interact with other users. The developers included a FOAF Pane into a Firefox extension called *Tabular*, whose last release was in 2008 and therefore is not compatible to current Firefox versions.

Evaluation

This project is seemingly not under development nor runnable since 2008, but it is a good example for alternative architectural patterns for POSNs. The framework is built very flexibly and allows applications to build new components. For example the FOAF file may include public keys to enable end to end encryption. But FOAF is more a proposition on how to manage distribution, it doesn't implement a native encryption, and it is not a usable prototype for a POSN.

Table 4: Criteria evaluation of FOAF

Mandatory Requirements	
Criterion	**Outcome**
End to end encryption	No
Data integrity	No
Usability	Low
Mobile friendly	No
Consistent	High
Available	Servers needs to be always online
Optional Requirements	
Criterion	**Outcome**
Open Source	No
Distributed	Multiple servers
Obfuscated Social Graph	No
Identity validation	Not natively

4.2.2 A Security API by Backes et al.

Features

In 2011, Backes, Maffei and Pecina presented a cryptographic framework to build POSNs with [30].

Each user is represented by a pseudonym. A pseudonym is created using a one-way function over a random value and used to authenticate a client using a zero-knowledge proof. This ensures that no other user may impersonate the pseudonym and that a client may prove to be some identity without the need to reveal it. The API provides basic tasks like establishing a friendship relation, posting a comment, and requesting an image. All requests are possible without the need to reveal the user's identity.

The API provides methods to establish a relation, receive file handlers, and to read or write to a file.

A relation may be direct (i.e. friendship relation) or indirect (i.e. friend of a friend). The indirect relation allows clients to use a specific friend to send a message to a mutually exclusive friend of this friend. Authentication may be provided revealing the pseudonym, using an already established relation, or anonymously. The authenticator may decline certain methods, for example in a common OSN an anonymous authentication might be unwanted.

All communication consists of either zero-knowledge proofs (bound to a receiver to prevent impersonation attacks) or encrypted messaging. This means all messages attackers may sniff are either with no value for them or encrypted.

Evaluation

The underlaying architecture needs a public key infrastructure and an anonymous channel, which both are not part of the framework. For key validation it is possible to create a key infrastructure based on a CA, a web of trust, or a manual verification system. For the anonymous channel the developers propose to use onion rings or mix nets. For a deeper analysis it should be measured how such an anonymous channel affects response times and if it slows down the network.

The lose coupling of those elements provides a very flexible framework for different use cases. The API may be used not only for POSNs, but for anonymous, private, collaborative forums.

Because access control is implemented in such a way that servers don't need to save their contact's pseudonym, a hacked server doesn't reveal the identities of the user's contacts. The network doesn't restrict the usage of pseudonyms, allowing a user to create and interact with multiple pseudonyms. This increases privacy since a user may use different pseudonyms for different use cases and interactions, which makes the user harder to trace and therefore increase privacy.

Every API call needs a zero-knowledge proof, which means those proofs need to be fast. The framework implements $\sum$ zero-knowledge proofs that only rely on three messages

(commitment, challenge, response [31]) and fast hashing algorithms.

Since the API calls are mainly handled directly between two communicating hosts, the consistency is high. But this also means that if the data owner is offline, availability is reduced. It is possible that calls via indirect relations may be used to increase availability.

Table 5: Criteria evaluation of Backes et al.

Mandatory Requirements	
Criterion	**Outcome**
End to end encryption	Yes; A public key infrastructure is needed
Data integrity	Depends on underlaying key infrastructure
Usability	Unknown
Mobile friendly	Unknown
Consistent	High
Available	Unknown; low to medium
Optional Requirements	
Criterion	**Outcome**
Open Source	No
Distributed	Multiple servers
Obfuscated Social Graph	Yes
Identity validation	Explicitly not intended

4.2.3 Lockr

Features

Lockr [32] decouples private information from other OSN features like content delivery. The developers want the user to be able to chose a trustworthy OSN provider to store their private data and still be able to use features of other OSNs. If the users don't trust any OSN provider to host their data they are able to set up a host themselves. The decoupling of data providers and OSNs means that they don't need to register on and input their data into different OSNs. For example, today users have different kinds of content like videos, photos, dating, or blogging. For those use cases they register on different OSNs where they need to enter all their private data again. With Lockr they can place their data on one trustworthy host and grant or limit access to OSNs. They may write a central access policy that is used by all OSNs to unify access control.

Parallel to the security API by Backes et al, Lockr uses zero-knowledge protocols. Lockr describes relationships via *social attestations*, which are signed XML files to describe a relationship such as 'friend'. A post is always related to a *social access control list* that defines the relationship to the publisher a reader needs.

That means that if person A is registered to one OSN and person B is registered to another OSN, they may receive each others' data only by providing the social attestation and are not required to register an account at the other OSN. When two users that don't know each other but assume they share a mutual friend want to speak with each other, of course without the need to reveal any of their information (zero-knowledge), they may use an encryption key bound to a relationship to that mutual friend. Only if they indeed share that friend they may decrypt each others' messages.

Lockr uses decentralized peer to peer systems such as BitTorrent to distribute content without CDNs. They implemented Lockr into an already existing BitTorrent client and created *social torrents*, a torrent that may only be downloaded when a correct attestation is provided.

Revocation is handled using three approaches: Through expiration dates, exclusion lists within the access control, and new attestations that state the new relationship.

Evaluation

Using a central server for hosting personal data and linking other OSNs to it is a concept already discussed in FOAF. But FOAF didn't specify how access control is handled, each server may implement access control itself. Lockr wants to specify unified access control based on zero-knowledge protocols. Both projects show that it may be a good idea to rethink about the general organization structure of current OSNs and that a lose coupling may increase privacy drastically. Using zero-knowledge protocols gives users the ability to establish a secure connection with each other just by having a mutual friend, a feature that is missing in FOAF (despite the name).

Besides the BitTorrent implementation, they implemented a prototype in the cen-

tralized OSN Flickr and tried to show that their system is very flexible. For Flickr they programmed a FireFox plugin that communicates with their proxy server. Flickr corresponded to the trusted OSN that holds private data, their server represented the content provider. But this design means that the OSN provider Flickr is still able to access every content of the user, which is not the desired result. For every photo a user uploads they upload a dummy photo publicly and the real photo set as private. Private photos are viewable only by having the appropriate link. Their proxy server would handle authentication and provide the hidden link only to clients that are allowed to view it. Because they couldn't implement the zero-knowledge proof at the client side, they sent the attestation directly to their server. This is a huge security violation since the proxy server may reuse the attestation to authenticate itself as the user. In result both servers, Flickr and proxy server, may violate or ignore the access policy of the user. In addition every authenticated user receives the hidden link and may distribute it using other channels. Therefore the implementation may be breached by every participant and is not considered to be useful.

Table 6: Criteria evaluation of Lockr

Mandatory Requirements	
Criterion	**Outcome**
End to end encryption	No
Data integrity	No
Usability	Unknown
Mobile friendly	No
Consistent	High
Available	Storage server needs to be always online
Optional Requirements	
Criterion	**Outcome**
Open Source	No
Distributed	Multiple servers
Obfuscated Social Graph	Yes
Identity validation	Handled by OSN

4.2.4 Persona

Persona [33] is a distributed network relying on a distributed storage system. It uses a hybrid encryption that consists of a ciphertext-policy attribute-based encryption (CP-ABE) and traditional public keys.

ABE gives the user the ability to create a policy that specifies attributes a reader needs in order to decrypt the data. Only when readers have the specified combination of attributes, they are able to decrypt the file. This allows the files to be stored publicly, even on untrusted servers. In Persona, attributes mainly represent groups like 'close friends', 'co-workers', or 'programmers'. Because ABE allows logical operators, a post may then be encrypted for attribute combinations, joined with logical operators or inequality signs. Attributes may also describe the key generation date. An example for an access relation:

$$keyYear < 2016 \vee (co\text{-}worker \wedge programmer)$$

The author of a post does not need to know who exactly is able to decrypt the data. It is possible to create a message and encrypt it for friends of a friend without actually knowing who exactly matches this access policy. This works if the group creator shares the group attribute names with the members of the group. It is even possible to give users a private key that matches the required attributes after the message was encrypted without the need to change the cipher text.

Every user creates an ABE public key and an ABE master secret key. Users may then categorize their contacts and therefore generate each friend a private ABE attribute secret key including the given attributes.

Persona was prototypically implemented using a Facebook app, but it used old Facebook app permissions and is not usable today.

Evaluation

Parallel to diaspora, sharing is asymmetrical, meaning only the sender defines a group of recipients, the recipients don't need to accept a membership. But unlike diaspora, Persona can ensure that only the recipients may read the file through encryption. Even when a server is compromised only the desired recipients are able to decrypt the files.

Since group management is a pattern already found in Facebook or diaspora, protecting it with a very similar encryption method makes the application very transparent, trustworthy and understandable. It ensures the data is really only readable by the configured recipients and not by the owners of the application.

Technically, parts of the features may be accomplished without ABE. A client may generate a key for each group (i.e. a symmetric key) and distribute it securely to the members of that group (i.e. using their public key). The key is cached on every participant's machine and messages may be exchanged securely. When a symmetric key is used, new users may be added easily to the group even after the message was sent. But

using ABE has two advantages. Firstly access control within ABE allows the author to formulate logical relations between attributes. When the sender decides to allow a subset of two groups (i.e. *co-worker* $\wedge$ *programmer*), this may be accomplished with ABE natively, while without it a new group needs to be created that only consists of that subset of recipients. Secondly sharing to third parties whose members are unknown to the sender (i.e. friend of a friend) is much easier to do with ABE than in a common public key environment.

Note that ABE features have a great cost since ABE is one hundred to a thousand times slower than RSA. The developers claim that they can minimize that overhead through a hybrid design, which means using symmetrical keys for each group and only using ABE operations when really needed.

The developers ported the framework to iOS. A performance measure showed on an iPhone 1 “decryption of ABE encrypted text fragments smaller than 1KB takes approximately 0.254 seconds”.

A problem is revocation: If a user is moved out of a group this means giving all remaining users a new key, which is, depending on the group size, a computational overhead. The removed user is still able to decrypt already existing files.

Table 7: Criteria evaluation of Persona

Mandatory Requirements	
Criterion	**Outcome**
End to end encryption	Yes
Data integrity	Yes
Usability	Medium
Mobile friendly	Yes
Consistent	Depends on underlaying storage service
Available	Storage Service needs to be always online
Optional Requirements	
Criterion	**Outcome**
Open Source	No
Distributed	Multiple servers
Obfuscated Social Graph	Unknown
Identity validation	No

4.2.5 The social network scheme proposed by Sun, Zhu, and Fang

Features

Sun, Zhu and Fang propose a POSN scheme [34] with a revocation concept. A data owner defines access groups like 'co-workers' or 'family' and which data sets are accessible for their users. The access granularity of those groups may be configured by the data owner. The group management is organized in a dynamic way, allowing the data owner to constantly add or remove users from a group without the need to create new keys for everyone.

The scheme relies on Public key encryption with keyword search, broadcast encryption (*BE*) and identity-based cryptography (*IBC*). Keywords are stored in an encrypted way on the server, allowing users to send encrypted keywords to the server and search for them in encrypted texts (*Trapdoors*). This means users may search for certain attributes server side without the need to download and encrypt every cipher text.

Parallel to Persona, a data owner may publish a post before adding recipients to the matching groups. The recipient group is fixed, not the members of that group. Unlike Persona, a user that was revoked is not able to decrypt any post anymore. However, if the users recorded keywords and the corresponding trapdoors they may search for those keywords and the server would even return new documents. Of course those users would be unable to decrypt those posts.

Evaluation

In contrast to Persona that uses ABE, IBC has easier revocation management. In ABE a user is part of many groups and a message is encrypted for a specific set of groups, joined with logical operators. If a group membership is revoked, the group owner needs to generate a large set of new keys. In IBC however, each user has only one unique key pair. Since the developers implemented BE, revocation only results in generating one new key. ABE has the feature that a post only needs to be encrypted once, where IBC needs to do multiple encryptions for each *role* (members of certain groups). The authors claim that this is not a big problem since an OSN user doesn't have much different groups and private data relies on a symmetrical and therefore fast encryption.

The scheme divides parties into trustworthy and untrustworthy ones. It assumes that the POSN provider and credential authority is a trustworthy party, whereas data is saved on an untrusted third party. It may be a wrong idea to trust POSN providers and may be that a malicious providers is able to breach the system security.

Table 8: Criteria evaluation of Sun et al.

Mandatory Requirements	
Criterion	**Outcome**
End to end encryption	Yes
Data integrity	Yes
Usability	Medium to High
Mobile friendly	No
Consistent	Depends on third party
Available	Depends on third party
Optional Requirements	
Criterion	**Outcome**
Open Source	No
Distributed	No
Obfuscated Social Graph	Unknown
Identity validation	No

4.3 Enhancing privacy within existing OSNs

4.3.1 flyByNight

Features

flyByNight [35] was a prototype of a Facebook app to encrypt messages. By the time this paper was written it was not accessible anymore, but it showed an important try on how to build encryption into Facebook.

flyByNight allows users to send one to one and one to many messages. All encryption and decryption takes place within the client so that Facebook only sees encrypted messages. The application uses an asymmetrical encryption via elgamal. Users may generate a key pair in the flyByNight app interface. The key is symmetrically encrypted with a chosen password and stored on the flyByNight servers.

The app was developed with usability in mind. The user is only required to remember one additional password and may use the service from other computers or even mobile devices (as long as they visit the Facebook desktop website). Encryption process and key generation take place solely in JavaScript, meaning no additional software or specific knowledge is required.

A Facebook user may have hundreds of contacts. As the developers realized that encrypting the message for every recipient takes too much time, they built a proxy cryptography server. This means the sender needs to encrypt the message only once and the proxy server helps encrypting the message for a specific recipient on demand. The proxy server is never able to decrypt the message by itself, therefore the application still uses end to end encryption. This proxy server may bring scalability issues when more users access it.

Within the app interface, the user selects one or many recipients from contacts using flyByNight. The application then generates the cipher text. A recipient may decrypt the message in the same way via the app interface.

Evaluation

One big problem with the app is that it is a Facebook app and therefore limited to the app bounds. It creates a separated interface to interact with and therefore is not embed into the main Facebook page, meaning the user always needs to switch between Facebook and the app. This is a big usability trade off, it is in question if users would still use it.

Because of how Facebook apps are technically realized, all communication to flyByNight is proxied by Facebook, giving Facebook the ability to attack the system. The developers argue that “using flyByNight to send information through Facebook may endow that information with legal privacy protection that compensates for its technical vulnerability”. As a safeguard the developers propose a browser plugin that may verify the JavaScript code so that security attacks would become immediately apparent to the

user. This does, however, not prevent possible attacks by manipulating keys within the database since database communication is proxied, too. In addition, the overhead of a plugin to check for security attacks would decrease usability even more.

flyByNight is not able to encrypt images, which today is a big disadvantage. The developers argued it was not possible to load images from disk into JavaScript, but today this is possible. It may, however, have performance issues since images are significantly larger than text messages.

Table 9: Criteria evaluation of flyByNight

Mandatory Requirements	
Criterion	**Outcome**
End to end encryption	Yes; Not images; Manual key validation
Data integrity	Yes
Usability	Low
Mobile friendly	No
Consistent	Depends on underlaying OSN
Available	flyByNight server must be always online
Optional Requirements	
Criterion	**Outcome**
Open Source	No
Distributed	No
Obfuscated Social Graph	No
Identity validation	Handled by OSN

4.3.2 NOYB

Features

NOYB [36], short for *None Of Your Business*, is a Firefox plugin with the approach of encrypting Facebook user profiles while being undetected by the OSN. Instead of encrypting the profile data of a user, it is mixed with the corresponding data of other users. NOYB splits a user profile in so called *atoms*. All atoms of the same class (for example the given name or the date of birth) are pseudo randomly mixed between the users, so that each user gets a "random" profile with legitimate data, making the obfuscation hard to track for the social network. If atoms were simply swapped, people would be able to know other user's atoms. That's why NOYB uses another approach: There are well known append-only dictionaries of atoms. When encrypting an atom the index of the atom is symmetrically encrypted, resulting in another pseudo random index. The cipher text is the atom at that new index. Friends are able to undo the encryption and therefore may see the original atom.

Evaluation

The basic aspect of mixing up information in such a way that the OSN cannot detect it easily is an approach not seen in much POSNs. The development team took measures to increase security through saving dummy data into the dictionary and creating dictionaries dependent on certain atoms like gender or age so that interests become more probable. Still the efficiency of the encryption scales with user count, therefore a smaller user base exploits more information about its users.

The application mixes up information like favorite books or TV shows. Whilst this works when a friend visits the profile and is able to reverse the process, it also means the Facebook news feed shows information on random topics. Changing the user's profile results in an impaired news feed and therefore reduces usability drastically.

Key exchange is not managed by the system, resulting in less usability and a bigger chance of user errors due to insecure communication channels. Another missing part is the encryption or obfuscation of messages. It is not enough to only encrypt user profiles and keep their messages and status updates in plain text, because the information in such messages must be considered at least as important as profile information. Since every atom must be saved within the dictionary, encrypting those message would increase the size of the dictionary very fast and it would result in making every private message public to everyone. The dictionaries may have scalability issues when the user base grows and serve as a single point of failure.

Table 10: Criteria evaluation of NOYB

Mandatory Requirements	
Criterion	**Outcome**
End to end encryption	Only profile; Manual key exchange
Data integrity	No
Usability	Low
Mobile friendly	No
Consistent	Depends on underlaying OSN
Available	NOYB servers need to be always online
Optional Requirements	
Criterion	**Outcome**
Open Source	No
Distributed	No
Obfuscated Social Graph	No
Identity validation	No

4.3.3 FaceCloak

Features

FaceCloak [37] gives users the ability to *shield* their data from other users or the OSN provider. FaceCloak has been prototypically implemented as a FireFox plugin and handles only Facebook accounts.

The user may turn encryption on or off for each information piece. To turn it on the user prepends the text with markers (implemented as '@@'), for drop down menus the application injects a second menu.

FaceCloak relies on a third party server that holds cipher texts. Each user may chose their third party server to store their data. There is a server provided by the authors and users may create a server using an open PHP / mySQL implementation. All communication handled by that server relies on TLS, therefore a user needs to have an SSL certificate.

When users post information, the application intercepts and encrypts marked data, sends it to the third party server, and posts fake data to the OSN. Decryption is handled in the same manner: FaceCloak receives the cipher text from the third party server, decrypts the post, and injects it into the site. Fake Data is generated by internal dictionaries (for surnames and names) and Wikipedia.

Each account generates a master key and a personal index key, which both are shared with the contacts. The master key is used for symmetrical encryption, decryption, and integrity validation of posts. The third party server stores information in a key/value store, where the key is calculated through a hash over the personal index key and an identifier for the information. The value is the encrypted information and a MAC for integrity validation.

If users update their own profile, they use their own keys, while when they post a message to a friend's wall they use that friend's key. This ensures that the friends of a friend that are not mutual may still decrypt the data.

For writing operations to the third party server the clients use a personal access key that is not shared with anybody else. This prevents contacts of a user to publish in the user's name and is a counter measure to DDOS attacks.

Encryption is implemented via AES-128 and SHA1 for indices, validation is implemented via a message authentication code.

Evaluation

Prepending input fields with '@@' is unusual, feels like a hack and decreases usability. It would have been a good idea to inject a drop down where the user may select the visibility instead.

Parallel to NOYB, FaceCloak generates seemingly valid data for the OSN in order to remain undetected. While NOYB only encrypted profile data, FaceCloak encrypts messages as well, which is a very important feature. But it has some disadvantages, too.

Key distribution is currently handled via email. Since emails are sent unencrypted, this is a major security risk. Another problem is the use of symmetric keys since it also means that revocation of a user is only possible by creating a new key and decrypting and encrypting every single message again. In addition a malicious or compromised user may share a key from a specific contact and therefore nullify the security of said contact, while the attacker's security remains intact. As a comparison in a public key scenario publishing the private key compromises the attacker's security, publishing a session key only affects a single message.

Note that revocation is currently not implemented in the prototype. Also a user can only publish to all of his contacts at once, there is no possibility to create multiple groups of recipients and share content only to a specific group. Therefore all contacts would be able to decrypt personal messages if Facebook made the message visible to them.

Since every write is saved with a personal access key and a change of that data is not possible without the correct key, users cannot change texts other user posted on their wall. They can, however, delete them through Facebook features. It is unclear on how the third party server is informed that a post was removed.

Because the OSN provider shouldn't be able to identify FaceCloak users, fake data is not marked in any way. To check if data is encrypted, the reader needs the two shared keys, so the OSN cannot identify FaceCloak at all, which is a nice feature, very similar to NOYB. But this mechanic also means that the application never knows what information is fake and what is real. So it calculates the index for every information and tries to download it from the third party server. This behavior produces much unneeded traffic and may slow down the application.

Since cipher texts are saved onto the third party application server and the server doesn't have the keys, it cannot read the data. It can't manipulate the data either, since FaceCloak implemented a validation mechanism. It doesn't even know what users it hosts, since the indices are hashed. The server may, however, perform a replay attack and replace content by old content that was legitimate before. The impact of such an attack is believed to be low. Another possible attack is a timing attack. This would lead to an identification of FaceCloak users, but not leak any data.

The latest FaceCloak implementation is 5 years old and doesn't work anymore.

Table 11: Criteria evaluation of FaceCloak

Mandatory Requirements	
Criterion	**Outcome**
End to end encryption	Yes; Unsecure key distribution
Data integrity	Only for author
Usability	Medium
Mobile friendly	No
Consistent	Depends on underlaying OSN
Available	FaceCloak server needs to be always online
Optional Requirements	
Criterion	**Outcome**
Open Source	Yes
Distributed	No
Obfuscated Social Graph	No
Identity validation	Handled by OSN

4.3.4 Scramble

Features

Scramble [38] is a Mozilla Firefox add-on allowing to encrypt and decrypt data. It is not bound to a specific OSN and may be even used on other websites. It uses OpenPGP with a simple user front end, allowing to generate keys, encrypt and decrypt messages, and synchronize the key chain with OpenPGP key servers. Encryption and decryption happens client side within the browser, supported by a Java runtime. For usability reasons, the developers decided to not use the OpenPGP web of trust and rather let the user decide to trust a specific key.

The development team claims to work on a chrome extension in order to reach more users.

Since cipher texts are longer than their plain texts and there are OSNs like Twitter with a very limited character count, the developers added an optional service called *TinyLink* which saves the cipher texts and returns a link to post on the OSN.

For access control users may add the public keys of their contacts to groups. While encrypting a message they can then select groups and contacts as recipients.

Evaluation

The installation of the browser plugin is not as trivial as one might think. The extension itself is installed quickly, but users are required to install JCEP which is not a trivial thing to do. If not configured properly, the add-on shows short cryptic error messages to the user, making it hard to track what is missing.

In order to paste an encrypted message, the user must copy and paste the plain text into the interface and select recipients. If the cipher text is too long for the OSN the users then past the content into the TinyLink service and finally send the link via the OSN, where they should configure the same recipients if possible. The recipients then click on the link, paste the cipher text into the extension GUI, enter their password and then are able to view the plain text. It should be obvious that this process takes very long time and decreases usability drastically.

And what happens to users who don't use Scramble? If the sender only sent it to users using Scramble (which takes him more time), it would be fine. If not, maybe because the OSN doesn't provide status updates to a limited group of recipients, the recipients would see encrypted texts or links to encrypted texts. If they were unaware of Scramble, this would be very confusing.

Table 12: Criteria evaluation of Scramble

Mandatory Requirements	
Criterion	**Outcome**
End to end encryption	Yes; No key distribution
Data integrity	Yes
Usability	Low
Mobile friendly	No
Consistent	High
Available	High
Optional Requirements	
Criterion	**Outcome**
Open Source	Planned
Distributed	No
Obfuscated Social Graph	No
Identity validation	Handled by OSN

4.4 Interim result

As an interim result, a table with all the twelve analyzed projects may be found in Appendix A. In order to provide a quick overview, each cell is colored, where green means the requirement is fully matched, yellow means that it is nearly fulfilled or that there are small drawbacks, orange means that the requirement is matched poorly, red means the requirement is not matched at all, and blue means it is unclear, dependent on other software, or valued to be neutral.

In section 2, RQ 3a asked if there is a POSN that fulfills the requirements defined in section 3. The evaluation of the given projects showed that there is neither a finished POSN nor a framework to fulfill our criteria and therefore the answer is *No*.

RQ 3b asked how to approach a POSN that fulfills the requirements. This thesis will now combine the defined requirements from section 3 with techniques observed in section 4 and propose a reference architecture.

5 Proposed architecture for POSNs

This architecture is a proposition on how to accomplish a POSN to fulfill the requirements. It is not intended to represent the only way on how to build a POSN, but an idea for a reference architecture.

Type of application

The analyzed projects may be categorized roughly into two groups. The first consists of applications that are built on top of existing OSNs (flyByNight, NOYB, FaceCloak, Scramble) and the second group consists of standalone POSNs or frameworks to build such (diaspora, SafeBook, PeerSoN, FOAF, Backes et al., Lockr, Persona, Sun et al.). Which one is better?

Attracting users with a freshly built, empty POSN is hard, because users know that their contacts aren't part of the community yet. In contrast, building a POSN on top of an already existing OSN means that users don't need to register to a new OSN. But building on top of an OSN without the support of the owners means working against their will and probably breaching their terms and conditions. As seen at Persona, flyByNight and NOYB, the application may break when the OSN updates its software or permissions, probably meaning that users are unable to communicate or use the POSN. In consequence, such applications need to be reevaluated periodically. If it breaks it needs to be fixed very fast in order to keep its availability.

Because the analyzed approaches to build privacy into existing OSNs either decreased usability, don't work anymore today, or both, this thesis proposes to build a standalone application. To ease switching to the new POSN, it should be considered to implement an importer that transfers data from existing OSNs into the new application.

End to end encryption and data integrity

This paper defined end to end decryption as mandatory, therefore it needs to be implemented.

NOYB and FaceCloak obfuscated their encryption in order to be untraceable by the OSN provider. Since this thesis proposes to implement a standalone application, this is not required.

The analyzed projects used various types of encryption, namely symmetrical encryption, public/private key encryption, attribute-based encryption, identity-based encryption, and broadcast encryption. Symmetrical encryption, as seen in FaceCloak, is the fastest of them. But the analysis of FaceCloak already mentioned that key management, creating groups, revocation, and preventing malicious sharing of keys create big problems with symmetrical keys. Therefore it is a bad idea to use symmetrical encryption only. Instead a hybrid encryption as seen in Persona, the API from Sun et al., and FaceCloak is advised.

Current OSNs give the user the possibility to search. Implementing an encryption that allows an encrypted search like Sun et al. did may improve the POSN in near future. But thinking about how mobile devices will develop, when a mobile device is able to cache more information on disk and provide an offline search, this may become obsolete. Therefore the encrypted search is not a proposed main feature.

Because users are already familiar with the usage of groups, it is recommended to implement an encryption that works in the same way. As elaborated this may be achieved using different techniques: Persona uses ABE, Sun et al. use IBC, and in section 4.2.4 this paper stated that the behavior may be achieved using common asymmetrical encryption, too. To evaluate what techniques to use, an implementations of each possibility and a cross platform performance measure is needed. This is enough work for another paper and therefore cannot be discussed any further.

Key distribution needs to be handled by the POSN in order to prevent distribution via insecure communication channels. Key validation needs to be easy, this thesis proposes to use QR codes to give the users the possibility to validate keys easily and securely. If a user validated some keys manually and those trusted users validated some other keys, a strong verification trust chain can be built and used. In the same way, users should be able to untrust certain keys, so that impostors can be spotted by their friends. When a user reads a friend request from an unknown identity he can then be presented with a trust or untrust chain.

A client always needs to check data integrity. The analyzed projects use asymmetrical signatures and message authentication codes. Since both techniques are valid cryptographic approaches, this thesis doesn't propose one exactly. The user should always be notified if data integrity validation produces a warning or an error.

Usability

The programmer cannot expect the user to know cryptographic details. All errors need to be explained in a simple and short wording. The application needs to be easy to use and work flows need to be efficient. As a negative example Scramble required the user to constantly use bowser menus and manually coordinate a third party TinyLink service. Another negative example is flyByNight that instead of integrating in the known Facebook work flow, it created a second interface to encrypt and decrypt data.

A diaspora pod may only be set up by a person with administrative skills and/or dedication. This is unacceptable. If the user is required to install something it should be straight forward and easy. If for example the application needs a database, a web server, and an SSL certificate, everything should come within the installation package or should be created automatically. The database and a web server may be installed and configured automatically. An SSL certificate may be generated and signed on the machine, possibly using a free CA.

The application should be transparent. Users should understand easily who is able to read their data and how to give or revoke access. A good example is the ABE encryption used in Persona that uses already known group management structures and maps it into cryptographic features.

Mobile Friendly

Only PeerSoN and Persona were implemented mobile friendly, where only Persona took performance tests for mobile. The mobile encryption times were acceptable even though the performance tests were run on hardware that is completely outdated today. As already stated, mobile devices are still developing rapidly in performance, therefore power should not be a problem in the future. But using browser plugins like FOAF, Lockr, NOYB, FaceCloak, and Scramble do is considered a bad idea since they are not runnable on mobile devices.

This thesis proposes to use techniques that are available on all platforms like JavaScript, that are natively supported through cross platform libraries (such as basic asymmetrical encryption) or to implement new libraries into the required platforms.

Today mobile clients cannot participate in a peer to peer environment like desktop computers can. Currently a mobile client always needs a desktop computer to rely on, but in the future this may change.

Distribution

Besides completely centralized projects, this paper presented two types of distribution. The first one is a completely distributed peer to peer system, where every user's machine connects and synchronizes with the network. The second one distributes the user data onto different servers, where every server may hold multiple users.

It might be easier to distribute the content onto different servers that are always online without creating a peer to peer network. This would help mobile clients because they may connect to those servers without the need to be part of the peer to peer network.

Since this thesis proposes to implement an end to end encryption, it is not necessary to distribute the content. Therefore it is not proposed if and how to distribute content.

Consistency and Availability

Except for SafeBook and PeerSoN, all discussed projects were highly consistent or relied on the consistency of their underlaying existing OSN. This is for a simple reason: All of those projects were either not distributed at all, or split the data in such a way that one server always handles certain data. If that is the case, the servers must be highly available.

This thesis didn't specify if a POSN should be distributed. If it was not, this would result in high consistency and high availability, but also in low scalability and high cost.

SafeBook and PeerSoN are distributed in a peer to peer manner and therefore need mechanics to regulate consistency and availability.

In SafeBook the data is cloned onto machines of close friends, so that if the data owner is offline, readers may contact those mirrors. Consistency and availability are highly dependent on the number of mirrors and on how clients are selected to become mirrors. Considering that close friends are normally geographically close to each other, it may occur that the mirrors are offline at the same time as the data owner, which would mean the data is not available at all.

In SafeBook real time interactions like the chat function are disabled if the user's machine is offline, which increases consistency, but decreases availability. But availability needs to be high, it is not acceptable that a messaging function stops working if the recipient is offline. If the messages were sent to a clone, there may be consistency issues. In PeerSoN a DHT serves as a puffer to cache messages if the user's machine is offline. As long as the DHT is always online, this increases both, availability and consistency. Therefore if the POSN was heavily distributed it may be a good idea to add such a puffer. Which puffer to use is in question since OpenDHT did not work well. This needs to be evaluated in other prototypes.

Obfuscated Social Graph

Even if content is encrypted, POSN providers may learn information through communication meta data. Obfuscating the Social Graph is a good concept to limit the possibility of those security breaches.

If the POSN was distributed in a peer to peer way, an attacker would need to monitor meta data from every client, which makes such an attack very complex. Therefore obfuscating the Social Graph in a peer to peer network may be omitted, but still is a

good privacy improvement. A good example is SafeBook which obfuscates the Social Graph using matryoshkas, giving each core anonymous entry points and communication routes. In the same manner an onion router or some other anonymized channel may be used, as seen in Backes et al.

If the POSN is not distributed in a peer to peer manner, obfuscating the social graph should be considered an important feature. As seen in Backes et al. and Lockr, zero-knowledge proofs are a good way to start. Additionally, Backes et al. give each user the possibility to create different pseudonyms in order to become more untraceable to eavesdropper.

This thesis proposes to implement zero-knowledge protocols and to evaluate if some anonymized routing is possible and fast enough.

Optional identity validation

This thesis already proposed key validation mechanics. They can be applied onto identity validation easily. Trusting and untrusting chains based on real life, QR code validation, and transparency for the users regarding trust chains allow the users to validate both, keys and identity, without a centralized authority.

Further this thesis proposes to build a transparent trusting network where each user may approve or disprove other identities, perform out of band validation such as validating checksums or photographing QR codes, and where each time a user interacts with an unknown party a trusting chain or untrusting chain is presented. The user should be encouraged to validate his close contacts.

6 Conclusion

This thesis showed how critical missing privacy in social networks, especially in Facebook, is. In order to protect the user from abuse, a POSN must be developed. This thesis defined six mandatory and five optional requirements to build such a POSN. It analyzed twelve potential POSNs generally and in regard to these requirements. It showed that none of those projects was able to fulfill these requirements and therefore proposed structures and techniques based on these requirements for an architecture to build a POSN.

There are still multiple propositions for some requirements, and therefore no "absolute architecture" was found. For future work the remaining propositions can be further analyzed regarding scalability, cost, performance, cross platform implementations, and implementation expense.

Appendices

A. Evaluation table

Mandatory Requirements												
Criterion	**Diaspora**	**SafeBook**	**PeerSoN**	**FOAF**	**Backes et al.**	**Lockr**	**Persona**	**Sun et al.**	**flyByNight**	**NOYB**	**FaceCloak**	**Scramble**
End to end encryption	No	Yes; Trust chains built on real life trust	Yes; Key distribution is not implemented	No	Yes; A public key infrastructure is needed	No	Yes	Yes	Yes; Not images; Manual key validation	Only profile; Manual key exchange	Yes; Unsecure key distribution	Yes; No key distribution
Data integrity	No	Yes	Yes	No	Depends on underlaying key infrastructure	No	Yes	Yes	Yes	No	Only for author	Yes
Usability	Usage: Easy Administration: Hard	Unknown	Unknown	Low	Unknown	Unknown	Medium	Medium to High	Low	Low	Medium	Low
Mobile friendly	Mobile optimized Web Page. No native application	No	Yes	No	Unknown	No	Yes	No	No	No	No	No

...Mandatory Requirements												
Criterion	**Diaspora**	**SafeBook**	**PeerSoN**	**FOAF**	**Backes et al.**	**Lockr**	**Persona**	**Sun et al.**	**flyByNight**	**NOYB**	**FaceCloak**	**Scramble**
Consistent	High	Unknown; eventual consistent?	Messages are delivered instantly or eventually	High	High	High	Depends on underlaying storage service	Depends on third party	Depends on underlaying OSN	Depends on underlaying OSN	Depends on underlaying OSN	High
Available	Pod needs to be always online	Not when host and close friends are offline	Medium to High	Servers needs to be always online	Unknown; low to medium	Storage server needs to be always online	Storage Service needs to be always online	Depends on third party	flyByNight server must be always online	NOYB servers need to be always online	FaceCloak server needs to be always online	High

Optional Requirements												
Criterion	**Diaspora**	**SafeBook**	**PeerSoN**	**FOAF**	**Backes et al.**	**Lockr**	**Persona**	**Sun et al.**	**flyByNight**	**NOYB**	**FaceCloak**	**Scramble**
Distributed	Multiple servers	P2P	P2P	Multiple servers	Multiple servers	Multiple servers	Multiple servers	No	No	No	No	No
Obfuscated Social Graph	Pod Administrator sees contacts	Yes	Unknown	No	Yes	Yes	Unknown	Unknown	No	No	No	No
Open Source	Yes	No	No	No	No	No	No	No	No	No	Yes	Planned
Identity validation	No	Not natively	Not natively	Not natively	Explicitly not intended	Handled by OSN	No	No	Handled by OSN	No	Handled by OSN	Handled by OSN

B. List of figures and tables

List of Figures

List of Tables

C. References

[1] D. M. Boyd and N. B. Ellison, "Social network sites: Definition, history, and scholarship," *Journal of Computer-Mediated Communication*, no. 13, pp. 210–230, 2008.

[2] J. Cannarella and J. A. Spechler, "Epidemiological modeling of online social network dynamics," *arXiv preprint arXiv:1401.4208*, 2014.

[3] Statista, "Leading social networks worldwide as of april 2016, ranked by number of active users (in millions)," April 2016. [Online]. Available: http://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

[4] facebook. (2016, Apr.) Data policy. [Online]. Available: https://www.facebook.com/about/privacy

[5] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel, "You are who you know: Inferring user profiles in online social networks," Max Planck Institute for Software Systems, Feb. 2010.

[6] M. Kosinski, D. Stillwell, and T. Graepel, "Private traits and attributes are predictable from digital records of human behavior," *PNAS*, vol. 110, no. 15, pp. 5802–5805, 2013.

[7] G. Wondracek, T. Holz, E. Kirda, and C. Kruegel, "A Practical Attack to Deanonymize Social Network Users," in *IEEE Symposium on Security and Privacy*, 2010, pp. 223–238.

[8] C. Sun, P. S. Yu, X. Kong, and Y. Fu, "Privacy preserving social network publication against mutual friend attacks," *Transactions on Data Privacy*, no. 7, pp. 71–97, 2014.

[9] B. Debatin, J. P. Lovejoy, A.-K. Horn, and B. N. Hughes, "Faceboook and online privacy: Attitudes, behaviours, and unintended consequences," *Journal of Computer-Mediated Communication*, no. 15, pp. 83–108, 2009.

[10] C. Bauer, J. Koronuvska, and S. Spiekermann, "On the value of information - what facebook users are willing to pay," *ECIS 2012 Proceedings, Paper 197.*

[11] Statista, "Facebook's revenue and net income from 2007 to 2015 (in million u.s. dollars)," 2016. [Online]. Available: http://www.statista.com/statistics/277229/facebooks-annual-revenue-and-net-income/

[12] K. M. Heussner, "Woman loses benefits after posting facebook pics," *abcnews*, 2009. [Online]. Available: http://abcnews.go.com/Technology/AheadoftheCurve/woman-loses-insurance-benefits-facebook-pics/story?id=9154741

[13] C. Lunt, "Authorization and authentication based on an individual's social network," U.S. Patent 9,100,400, 2015.

[14] D. Kaye, "Promotion and protection of all human rights, civil, political, economic, social and cultural rights, including the right to development." Human Rights Council, 2015.

[15] J. D'Onfro, "Here's how much time people spend on facebook per day," Aug 2015. [Online]. Available: http://www.businessinsider.com/how-much-time-people-spend-on-facebook-per-day-2015-7?IR=T

[16] K. Dreyer, "Mobile internet usage skyrockets in past 4 years to overtake desktop as most used digital platform," comScore, Tech. Rep., 2015.

[17] A. Yarmula, "Strong consistency in manhatten," May 2016. [Online]. Available: https://blog.twitter.com/2016/strong-consistency-in-manhattan

[18] H. Lu, K. Veeraraghavan, P. Ajoux, J. Hunt, Y. J. Song, W. Tobagus, S. Kumar, and W. Lloyd, "Existential consistency: Measuring and understanding consistency at facebook," University of Southern California, Facebook, Inc., Tech. Rep., October 2015.

[19] H. Jaap-Henk and B. Jacobs, "Increased security through open source," *Communications of the ACM*, vol. 50, no. 1, pp. 79–83, January 2007.

[20] B. Witten, C. Landwehr, and M. Coloyannides, "Does open source improve system security?" *IEE Software*, vol. September/October 2001, pp. 57–61.

[21] C. Payne, "On the security of open source software," *Info Systems J*, no. 12, pp. 61–78, 2002.

[22] J. Leber. (2012, May) The biggest cost of facebook's growth. [Online]. Available: https://www.technologyreview.com/s/427941/the-biggest-cost-of-facebooks-growth/

[23] A. Shakimov, A. Varshavsky, L. P. Cox, and R. Cáceres, "Privacy, cost, and availability tradeoffs in decentralized osns," SIGCOMM WOSN, Tech. Rep., 2009.

[24] Diaspora. [Online]. Available: https://diasporafoundation.org

[25] *Federation protocol overview*, diaspora. [Online]. Available: https://wiki.diasporafoundation.org/Federation_protocol_overview

[26] Diaspora blog. [Online]. Available: https://blog.diasporafoundation.org/

[27] L. A. Cutillo, R. Molva, and T. Strufe, "Safebook: A privacy-preserving online social network leveraging on real-life trust," *IEE Communications Magazine*, pp. 94–101, December 2009.

[28] S. Buchegger, D. Schiöberg, L.-H. Vu, and A. Datta, "Peerson: P2p social networking: Early experiences and insights," in *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, ser. SNS '09. New York, NY, USA: ACM, 2009, pp. 46–52.

[29] C.-m. A. Yeung, I. Liccardi, K. Lu, O. Seneviratne, and T. Berners-Lee, "Decentralization: The future of online social networking," W3C, 2008.

[30] M. Backes, M. Maffei, and K. Pecina, "A security api for distributed social networks." in *NDSS*, vol. 11, 2011, pp. 35–51.

[31] I. Damgard, "On σ-protocols," *Lecture notes for CPT*, 2002.

[32] A. Tootoonchian, S. Saroiu, Y. Ganjali, and 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.

[33] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin, "Persona: An online social network with user-defined privacy," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 135–146, Aug. 2009.

[34] J. Sun, X. Zhu, and Y. Fang, "A privacy-preserving scheme for online social networks with efficient revocation," in *INFOCOM, 2010 Proceedings IEEE*, March 2010, pp. 1–9.

[35] M. M. Lucas and N. Borisov, "Proceedings of the 7th acm workshop on privacy in the electronic society," in *flyByNight: Mitigating the Privacy Risks of Social Networking*, 2008.

[36] S. Guha, K. Tang, and P. Francis, "Noyb: Privacy in online social networks," in *Proceedings of the First Workshop on Online Social Networks*, ser. WOSN '08. New York, NY, USA: ACM, 2008, pp. 49–54.

[37] W. Luo, Q. Xie, and U. Hengartner, "Facecloak: An architecture for user privacy on social networking sites," in *Computational Science and Engineering, 2009. CSE'09. International Conference on*, vol. 3. IEEE, 2009, pp. 26–33.

[38] F. Beato, M. Kohlweiss, and K. Wouters, "Scramble! your social network data," in *Privacy Enhancing Technologies: 11th International Symposium, PETS 2011, Waterloo, ON, Canada, July 27-29, 2011. Proceedings*, 1st ed., ser. Lecture Notes in Computer Science 6794 Security and Cryptology, S. Fischer-Hübner and N. Hopper, Eds. Springer-Verlag Berlin Heidelberg, 2011, pp. 211–225.