Concealed Communication in Online Social Networks

Fabian Schillinger(✉) and Christian Schindelhauer

Computer Networks and Telematics, Department of Computer Science, University of Freiburg, Freiburg im Breisgau, Germany
{schillfa,schindel}@tf.uni-freiburg.de

Abstract. Online social networks are used frequently: staying in contact with friends and sharing experiences with them is very important. However, users are increasingly concerned that their data will end up in the hands of strangers or that personal data may even be misused. Secure OSNs can help. These often use encryption to keep the communication between the participants incomprehensible to outsiders. However, participants in such social networks cannot be sure that their data is secure. Various approaches show that even harmless-looking metadata, such as the number of contacts of a user, can be evaluated to draw conclusions about a user and the communication. These attack methods are analyzed and existing secure OSNs are examined, whether these attack methods can be utilized to violate the user's privacy. To prevent these privacy attacks, protocols for a secure centralized OSN are developed. Metadata is obscured in the presented OSN and end-to-end encryption is used for secure communication. Additionally, communication channels are concealed like in mix networks such that adversaries cannot determine which user is accessing which data or which user is communicating with whom even with full access to the server.

Keywords: Online social networks · End-to-end encryption · Privacy · Security · Metadata · Attack · Mix network

1 Introduction

Many online social networks (OSNs) vie for the users' favor. They offer different unique selling points to make the user experience as good as possible so that users spend a lot of their time in these networks. Because, in many OSNs, money is earned through advertising, the more users are on the network and the longer they stay there, the greater the advertising income. These can be slightly increased even further if personalized advertising is displayed [14]. This is tailored to the respective user and increases the likelihood that they will react to the advertising and thus increases the profit of the network operators. In doing so, users can lose their anonymity. Personalized advertising is possible by evaluating profiles, contacts, or reactions to messages in social networks, etc. These practices are a thorn in the side of many users. That is why they are increasingly

B. Chen and X. Huang (Eds.): AC3 2021, LNICST 386, pp. 117–137, 2021.
https://doi.org/10.1007/978-3-030-80851-8_9

using privacy-friendly social networks or those they consider privacy-friendly, such as Telegram [25]. Furthermore, it is often not only the operator of a social network that analyzes the personal data. Also third parties, analyze the data they have access to. Often, apps or games in OSNs or code in displayed advertisements can gain such access. Third parties can do harmless things with the data, but they can also display malicious behavior and misuse the collected data for personal theft, for example. This may cause a rethinking of many users which leads to them registering in social networks that supposedly protect their privacy. This often is achieved by using encrypted communication. But privacy can be attacked and the sovereignty of the users over their data undermined no matter, whether the communication is encrypted or the user profiles are protected, but the metadata is accessible and analyzed. Research shows, that the possibility of privacy leakages through metadata should not be taken lightly.

Our Contribution

Our contribution consists of: first, a summary of possible privacy leakages in privacy-preserving online chats and online social networks. Second, an analysis of privacy-preserving OSNs is given, according to the privacy leakages. Third, following these findings, protocols are presented to create concealed channels between participants in an OSN, which relies on a client-server architecture. These concealed channels work comparable to a mix network. The channels provide end-to-end encrypted communication between two or more participants and further do not leak any evaluable metadata to the service provider of the OSN or another attacker. With concealed channels different possibilities are presented to provide the functionalities, an OSN should provide. These are, for example, profiles of the users, private messages between two or more participants, and discussion groups. A prototype of the scheme using a C# server and a HTML and JavaScript frontend is used to evaluate the impact on run-times when using the cryptographic algorithms. The presented approach of a secure and privacy-preserving OSN is analyzed, whether the possible privacy leakages are prevented.

Organization of the Paper

The paper is structured as follows: Sect. 2.1 summarizes different approaches to analyze metadata in online social networks and encrypted communication channels. Based on these works, possible leakages are displayed in Sect. 2.2. Following, in Sect. 2.3, different protocols for privacy-preserving online chats and online social networks are analyzed, whether they are secure against the leakages. In Sect. 3, protocols are presented to achieve secure communication. These protocols are used to construct a privacy-preserving OSN with encrypted communication. For the presented OSN, then, privacy-preserving functionalities are analyzed and compared to the previously found leakages in Sect. 4. Finally, Sect. 6 concludes the work and gives an outlook on future work.

2 Related Work

In the following, different possibilities to undermine user privacy, using metadata are collected. Then, privacy-preserving online social networks are analyzed, whether they are prone to these privacy leakages.

2.1 Privacy Analysis in Online Social Networks

The attributes users post in their profiles in an OSN can be used to predict the attributes of other users, according to [16]. The dataset of the Rice University network and the New Orleans Facebook dataset are used. One of the findings is, that friends are likely to share common attributes and these common attributes form groups. Using these findings, on the one hand, it is possible to predict the social circles of users. Using the social circles it is, on the other hand, possible to predict attributes of users.

In [23] two datasets of the OSNs Flickr and Last.fm, and in [1] three datasets of Flickr, Last.fm, and aNobii are analyzed to predict social links. In these OSNs, users can communicate, form groups of similar topics, create links to other users, and use tags to annotate content, such as shared pictures. Various observations are made: users that have more contacts tend to be more active regarding tagging and membership of groups, assortative mixing of nodes in the OSNs is detected, and different patterns of topic similarity between neighbors are found. For assortative mixing, i.e. the observation, that nodes tend to be linked to nodes with similar properties, various properties are investigated. For example, the degree of nodes, especially the nearest neighbor's degree, average number of tags of nearest neighbors, or average amount of groups. Using these observations similarity between users is calculated to predict social links with high accuracy.

Privacy leakage through metadata of decentralized OSNs is examined in [10]. Adversaries with distinct possibilities to access the data are considered. In a centralized setting, the service provider combines all of them. From the metadata of content, different observations are possible. The size of an object is an indicator for the type, as text messages are smaller than images or videos. From the structure of a group of elements conclusions can be drawn, e.g. amount of images in a shared album. The modification history of an object can reveal information, e.g. about user status updates or intensity of activity. Other information can be obtained from access control mechanisms, like encryption headers. Here, header sizes can allow estimating the number of encryption keys. Adding encryption keys or revoking them, can lead to re-encryption of contents and can allow drawing conclusions about changes in the relations between users. From re-using the same key for different objects one can learn about overlapping access rights. The communication flow can lead to more information. From tracking IP addresses, conclusions about online times or working habits can be drawn, using geo-IP mapping services routes, locations, and traveling information can be tracked. Access logs from shared content can be used to determine user-groups, ownership, and access patterns. Timing information may be obtained from newly

created objects and (re-)distribution of keys. Different control-operations, like login, adding friends, or searches can be observed.

Metadata of Twitter is analyzed in [19]. A tweet contains 144 fields of metadata. This allows drawing conclusions about the owner of such a post, without considering the actual content. Using machine learning approaches owners of posts can be identified from a group of 10.000 users with 96.7% accuracy.

Identifying social circles from network structure and user profile information is the task in [15]. Using machine learning approaches, a model is created that accurately identified social circles in Facebook, Google+, and Twitter.

Patterns in user behavior can be recognized even with encrypted traffic. In [5] encrypted traffic of different sequences of actions are collected for popular Android apps. A sample sequence for the Facebook app is to tap on the button to write a post, fill the textbox with some random text, and post this message. Analyzing the network flow as a set of time series it is possible to predict different patterns even with TLS/SSL encrypted traffic.

2.2 Privacy Leakages

From the findings in Sect. 2.1 the following list summarizes possible problems when metadata can be accessed in an Online Social Network where communication is end-to-end encrypted. Some of the problems alone may not necessarily lead to privacy leakages, but the combinations of different problems can lead to severe violations of privacy.

Structure of Network (*NS*)

- Degree of Node (*NS1*). How many contacts does a user have?
- Neighbors of Node (*NS2*). How many contacts do the contacts of a user have?

Structure of Data (*DS*)

- Count of Objects, Groups, Keys, ... (*DS1*). How many contacts, posts, messages, ... does a user, group, ... have?
- Common Objects, Groups, Keys, ... (*DS2*). Which contacts, posts, comments, ... are common between users, groups, ...?
- Size of Objects or Groups (*DS3*). What is the type of an object (text, media, ...), how many users are in a group?
- History of Objects or Groups (*DS4*). How often does an object or group change?

Timing (*T*)

- Time Series Pattern (*T1*). Are objects or keys accessed in a specific order, especially with a specific timing?
- Timing for Creating Objects and Distributing Keys (*T2*). Which keys are associated with which objects?

– Key Distribution and Re-Encryption (*T3*). Which keys belong to with which objects and which keys are created or deleted when objects are changed?

Control Information (*CI*)

– IP Address Logging (*CI1*). Track user behavior when the same IP accesses content, or different IP addresses access the same content, especially when the time of day is the same across different days.
– Geo-IP Mapping (*CI2*). Where is a user, which locations are associated with a user?
– Access Logs for Objects, Groups, or Ownership (*CI3*). Which user accesses which objects, groups, ...?
– Control Messages and Queries for Login, Friend Request, or Searches (*CI4*). What is a user doing in the OSN?

2.3 Privacy-Preserving Online Social Networks

todo. Various schemes introduce encryption of messages between two or more participants. Some of the schemes are designed for emails, others are designed especially for online message services. Most of the schemes rely on a client-server structure, but there are peer-to-peer approaches, as well:

A scheme for encrypted online chats is *Off-the-record* (OTR) [18]. The scheme uses new session keys k_i for each message i. Each key is negotiated through a Diffie-Hellman key exchange between two participants A, B with keys x_{Ai}, x_{Bi}, where keys x_{zi}, x_{zj} for $z \in \{A, B\}, i \neq j$ are independent. This introduces perfect forward secrecy for the communication. Possible leakage vectors could be: *DS3* and *T1* because an adversary could track the sizes and sending times of messages. When a dedicated server is used for the communication, additionally, *NS*, *CI1*, and *CI2* can be exploited by the server.
A peer-to-peer system for end-to-end encrypted messages between two participants was *Silent Circle Instant Messaging Protocol* (SCIMP) [17]. Elliptic Curve Diffie-Hellman was used to agree on a shared key for encrypted messages. As SCIMP was a peer-to-peer approach all leakages utilizing a server are mitigated, still the leakage vectors *NS*, *DS* and *CI* could be exploited by a service provider or another malicious relay.

Private Facebook Chat (PFC) [21] introduces end-to-end encrypted chats inside Facebook. The key distribution works by dedicated servers, that use the Facebook authentication mechanisms. Nearly all leakage vectors seem exploitable, except for *T2* and *T3*. No matter, whether the Facebook servers or the PFC servers are considered because an adversary on either of the servers can access all metadata.

Multiple peers can communicate securely using the approach described in [11]. A modified Diffie-Hellman protocol is used to find a common secret for the participants with the help of a server. The leakage vectors *NS*, *DS* and *CI* can possibly be exploited, because a server is used to manage the communication and keys.

A comparable approach is discussed in [30]. Here, elliptic curve Diffie-Hellman is used for the key agreement. Therefore, the same leakage vectors *NS*, *DS* and *CI* can possibly be exploited.

In the *Signal* protocol [13] a shared secret between participants is derived from a key chain. The inputs for this chain are found through a modified Diffie-Hellman key exchange. Messages are encrypted and new keys are used for every message. Still, the leakage vectors *NS*, *DS* and *CI* are possible, when the server is attacked.

Threema [26] allows end-to-end encryption of messages. Communication between two peers is encrypted using a shared key. Messages in groups are encrypted for each peer individually, using their public keys. Larger files are encrypted using a symmetric key, which is encrypted with each participant's public key. When the server is attacked, all leakage vectors *NS*, *DS*, *T*, and *CI* could be exploited.

Another end-to-end encrypted online chat is presented in [24]. Messages are encrypted using symmetric AES encryption. The necessary keys are encrypted for all participants of a chat using their public RSA keys. A chat can contain arbitrarily many participants and the number of participants can be changed, then new AES keys are generated and distributed. There are various possible leakage vectors when the server is attacked: all from *NS*, *DS*, *CI*, and *T*, because all necessary data to manage the OSN are stored in plaintext on the server.

Pretty Good Privacy (PGP) [9] and *S/MIME* [22] are methods for the encryption of emails. In both, public-key encryption is used to encrypt a symmetric key for every recipient of an email. The content of an email is encrypted, using the symmetric key. All following answers use the same keys. The main difference is in the verification of public keys: PGP constructs a trust system between participants, whereas S/MIME uses X.509 certificates. Possible leakage vectors are: attNetwork and *DS*, because recipients are known in plaintext. Leakages from *CI* are possible, if the adversary is one of the involved mail servers.

Other approaches introduce privacy into Online Social Networks:

FlyByNight [12] introduces client-side based encryption of content for Facebook. Each user has a password that is used to encrypt a private key for the key database of the system. Messages between participants are encrypted using their public keys. Proxy cryptography is used when more than two participants communicate. As flyByNight is an extension to Facebook all leakage vectors could be used when considering their servers. When considering only the servers of flyByNight, still, *NS*, *T*, *CI*, *DS1*, and *DS3* could be used, because the server manages all keys and messages. A decentralized OSN is *Safebook* [6]. Social circles from the real-life are used to construct trust relationships. Each node is surrounded by those structures, which are called matryoshkas. This is used to provide data storage and communication privacy. Another layer is a peer-to-peer network that enables application services, such as lookup. The internet is the transport layer in the scheme. Because of the peer-to-peer approach, most of the leakages are prevented, or at least very unlikely. E.g. to exploit *NS*, *DS*, or *T*,

trusted peers have to attack the user. Attacks through *CI* are unlikely because of the peer-to-peer structure.

In [7] another decentralized OSN is presented. Again, real-life trust relationships are utilized for trustworthy connections within the network. Multihop routing between trusted peers is used as an anonymization technique. Privacy leakages are unlikely, because, for *NS*, *DS*, or *T* trusted peers have to attack the user.

In the OSN *Persona* [2] users define who can access their information. Attribute-based encryption is used to share secrets within groups of participants that have at least one attribute in common. Further, each user owns a key-pair, such that the public key can be used to encrypt content specifically for this user. Although communication data is encrypted, various usable metadata may accumulate on the server, therefore, exploits of *NS*, *DS*, *T*, and *CI* seem possible.

Snake [3] is an OSN which is written in HTML5 and JavaScript. It uses the WebCrypto API to encrypt messages between peers. One can not conclude which peers communicate, because addresses are masked inside the database. When users establish a friendship they agree on a shared key and addresses to send and receive messages. These addresses change with every new message. Therefore, exploiting *NS* and*DS* seems to be not possible, when considering communication between peers, but *T* and *CI* could be exploited. When a user logs in the server has to prove the necessary encrypted data and exploits of *DS1*, *DS3*, and *DS4* can be possible.

Vuvuzela [28] is a system to ensure private messaging. Most of the metadata of communication is hidden by the system. The participants use locations, called dead drops, to place messages for other users. Managing messages is performed in a round-based approach by multiple servers. A user sends its request messages, like placing messages to or retrieving messages from dead drops to a server. After a round, the collected operations are performed by the server. Then, dead drops are discarded and cannot be accessed in further rounds. All messages that are not delivered, are deleted. Privacy is preserved through different measures: first, a constant amount of messages and sizes of messages for each user per round is set, where messages are delayed to other rounds or empty messages are introduced. Second, the servers work as a mix network to anonymize dead drops. Each message is encrypted for the next server, such that no server can determine, which message comes from which client. The architecture prevents most of the attacks, the only observable information is the number of participants in a communication (*DS3*). Further, the protocol only ensures messages between users, other desired information exchange of OSNs is not supported. This includes all persistent information, like user profiles or groups.

The protocol *Stadium* [27] allows private messaging, comparable to Vuvuzela. Here, messages are, again, exchanged via dead drops and a mix network is set-up between the senders and recipients of messages. The protocol reduces the amount of noise needed in the network, to keep the communication private, compared to Vuvuzela. Still, persistent information cannot be stored, using the protocol.

Another protocol that utilizes mix networks is *Loopix* [20]. Again, no persistent information is stored, apart from the information that is temporarily stored on so called providers, when the recipient is offline. But, as soon, as the information is delivered it is removed from the provider, as well.

2.4 Mix Networks

In [4] mix networks are presented. A mix network is a routing protocol where a message is sent to a proxy, called mix, that forwards the message to another mix or the recipient. When layered encryption is used between the participants the path of a message becomes hard to trace. This allows achieving anonymity of the sender, of the recipient, or both. Consider a network with sender S, recipient R, and mixes $M_1, M_2, \ldots, M_n$ with public encryption keys $s, r, m_1, m_2, \ldots, m_n$ and a message N.

Anonymity of the sender is achieved by encrypting the message to $n = \{\{\{\{N\}_r\}_{m_n} \ldots\}_{m_2}\}_{m_1}$. n is sent to M_1, who decrypts the message and forwards it to M_2, and so on. Finally, R receives $\{N\}_r$ from M_n. Because each recipient of the message only knows the participant before, and after, the flow of the message is concealed and R does not get to know that S is the sender.

The anonymity of the recipient is provided when the recipient generates a return address. The return address is an encrypted sequence of mixes that the sender has to use, where only the first mix is known to the sender. The recipient sends this sequence to the sender, via the mix network.

By combining both schemes, the sender and the recipient of a message can remain anonymous to each other.

Different methods are needed, that outsiders cannot track messages through a mix network, such as removing of duplicate messages. This would allow an attacker to find a connection between a received message and the next mix because the duplicate has to be sent to the same receiver. Further, messages have to be modified by a mix to prevent comparing incoming with outgoing messages. Additionally, messages at a mix have to be collected and either forwarded at random or together with other messages and the forwarding procedure has to produce a different ordering of messages.

3 Proposed Protocols

The following protocols ensure private communication within an Online Social Network (OSN), based on a client-server architecture. No metadata is leaked when users communicate. This is achieved through concealed communication channels.

3.1 A Concealed Secure Channel

In a client-server model, a concealed channel between two clients uses the server. Such a channel can be created when one of the clients has a concealed channel

to an address on the server, which is known to the other client and accessed through a concealed channel. This address is called *concealed address*.

Definition 1 (Concealed Address). *A tuple* (c, p_R, p_W, p_O)*, where* c *is an unique address at the server is a concealed address.* p_R*,* p_W*,* p_O *are values to prove that one is allowed to read or write messages to this address, or that one is the owner of the address.*

A *concealed address* can be created by sending a CreateAddress-Message to the server. The message contains three corresponding values, called *address keys*: p_R, p_W, and p_O. The *concealed address* provides an address on the server where clients can read messages from or write messages to, when the according proofs are provided. Another proof determines the ownership of the *concealed address*. The *address keys* are used to provide these proofs.

Definition 2 (Address Key). *An address key is a value* p_R*,* p_W*, or* p_O*. For reading or writing messages, or for proving ownership of an address. Each address key is a public-key of a cryptographic protocol for signatures. An address key can have a wildcard-value* $*$*.*

Clients can prove that they are allowed to read messages from the *concealed address* by using the p_R value, which is called *read address key*. p_W, the *write address key*, is used to verify if a client is allowed to write messages to it. It can be proven that the client is the owner using the *owner address key* p_O. For the verification, the message is signed by the client using the corresponding private-key. To ensure enough entropy in a message, a (signed) random nonce is always part of the message. Then, multiple (similar) requests from the same *concealed address* always have a different signature. This prevents replay-attacks. When using secure cryptographic protocols, the private-key cannot be computed from the public-key. Therefore, providing the correct signature to a message proofs the knowledge of the private-key. This ensures, that a client is allowed to read or write messages, or modify a *concealed address*. The proofs are only a verification against the server. Messages stored at a *concealed address* can be encrypted using a *content key*. Without knowledge of the *content key* no client can decrypt the messages from a *concealed address*.

Definition 3 (Content Key). *A content key is a value* k*, which is used as a key for the encryption and decryption of messages stored at a concealed address.*

The *content key* normally is a symmetric key, known to all participants because of the increased speed compared to public-key cryptography. However, when no symmetric key is established, one of the participants can choose it and encrypt it once with the public-key of each participant. In this case, there are multiple *content keys*. Further, all messages at a *concealed address* can have different *content keys* or are not encrypted, at all. The *content key* and *address keys* are independent from each other. They can be negotiated between the participants or distributed in person and later changed between messages. Using multiple *concealed addresses* concealed communication channels, comparable to a mix network (see Sect. 2.4), can

be established. A participant of the OSN, that wants to provide a mix network generates multiple *concealed addresses*, where the *write address keys* are $*$. This allows everybody, knowing one of the addresses, to write messages to it. When the messages are encrypted with, either the public key of the mix or a symmetric key, only known to the mix and the sender, they can be decrypted by the mix, only. These messages, then, contain another encrypted message for another mix or a recipient and an address, where it has to be written to. Messages are collected by the mix, decrypted, reordered, merged if possible, and then written to the specified addresses. Messages can be merged when they have the same target address. Then, they are concatenated and written as a single message. For increased security, duplicate messages are deleted. Using this construction, a communication between a sender and a receiver can be established, where the sender is anonymous, or the receiver is anonymous, or both are anonymous to each other. To reduce the number of messages stored to the server, mixes delete messages, after they are forwarded. However, to prevent attacks with duplicate messages, the mixes have to wait a certain amount of time, depending on how many messages they receive and have to forward. To further decrease the storage needed, the digest values of messages can be stored instead. Together with a sliding limited window of acceptance replay attacks can be prevented. Here, such a window only allows messages that are not older than a specified timespan. When a path is used that achieves anonymity of the sender, this anonymous sender can prove ownership of an address to the server. This allows participants of the OSN to exchange addresses. E.g. a user A can create a *concealed address* c_A with some *address keys*. A can send the *owner address key* via a concealed channel to B. Then, B can prove ownership of the *concealed address*, via another concealed channel. Exchanging *concealed addresses* between users and mixes cannot be tracked by the server or another user (Fig. 1).

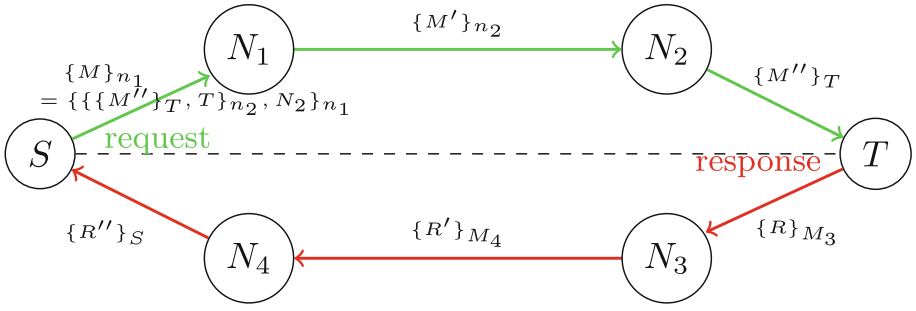

Fig. 1. An exemplary message flow from sender S to receiver T over the nodes N_1 and N_2. $\{M\}_{n_1}$ can be decrypted by N_1 and contains the target address of N_2. Finally, message $\{M''\}_T$ is received by T and decrypted. The return path across N_3 and N_4 is encrypted inside the message, such that T does not need to know the addresses of N_3, N_4, or even S. I.e. M'' is a tuple $(m, N_3, e_{n_3}, \{(N_4, \{S\}_{n_4})\}_{n_3})$, where m is the message for T, e_{n_3} is the public-key of N_3, and the last ciphertext contains the return addresses for nodes N_3 and N_4. These addresses are encrypted and can only be accessed by the respective node.

3.2 User Information and Postings

Sharing personal user information, like the date of birth or residence, is part of social networking. In many OSNs users can create profiles to share such user information. In our approach, it is possible to create a user profile using *concealed addresses*. The user can decide, how the profile is stored at the server: a user profile can consist of a single *concealed address*, containing all the information a user wants to share. This information can be encrypted or stored as plaintext at the *concealed address*. The user then decides who receives the *concealed address*, the *read address key*, and, in case of encrypted information, who receives the *content keys*. Storing the user profile as a whole at the server is possible but not recommended: on the one hand, a user often has to update the whole profile or re-encrypt the contents, when some information or access rights are changed. On the other hand, defining fine-grained access rights is complicated. A user can share some of the information with everybody in public, while other information is shared with specific contacts, only. To achieve this, a user has to distribute multiple different keys. It is recommended that the user creates a *concealed address*, where the *read address key* is $*$. This *concealed address* is used to store all addresses of the profile information, like a *concealed address* containing the public keys of the user, so other users can verify signatures or send encrypted messages, another *concealed address* for the date of birth, another *concealed address* for the residence, and so on. Then each linked *concealed address* can have unique *address keys*. To make the tracking of user information through linked addresses difficult, each linked address itself can contain another linked address, which is encrypted. This prevents an adversary from learning which addresses are connected.

A user may want to share postings. These can contain the personal experience, for example, pictures from the last holidays. This information can be stored at the user profile, like personal information, or the user can link a single, or multiple *concealed addresses*. Each address can contain a list of posts. This allows the user to create different information feeds, where each feed can have different keys, and therefore different access rights.

The construction of a user profile using *concealed addresses* allows not only to post personal profile information and user posts, but any type of information: the user may share a calendar with fine-grained access control over each appointment, different blogs or vlogs, picture albums, or wikis. The user can create different *concealed addresses*, where the *write address keys* are $*$. These addresses work as different pinboards, where other users can post messages visible to the user and everybody knowing the correct *read address keys* and *content keys*. Further, the user can use private *concealed addresses* to store notes, bookmarks, or other private information, like passwords. For every information, the user can create a new *concealed address*, a new *address key*, and *content keys* to be able to define the access rights every time. Using *concealed addresses*, the user can create collections of keys. These collections can be shared with single users or with groups of users. An exemplary user profile is displayed in Fig. 2.

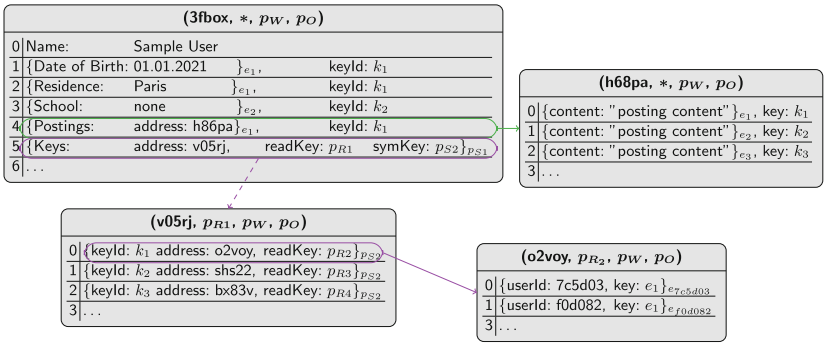

Fig. 2. A user profile solely relying on *concealed addresses*. The construction allows a user to hide its personal information, except for the name. The fields are encrypted using keys with Ids $k_1, \ldots, k_3$, which are stored in a *concealed address*, which is not directly, i.e. in plaintext, linked from the profile. When a user receives the key p_{S1}, it can decrypt the entry containing the keys. Then, the *concealed address*, with the keys can be accessed. Each key is encrypted, using a user-specific key. Using these keys, the fields from the profile can be decrypted and their *concealed addresses* can be read.

3.3 Personal Messages and Group Messages

Personal messages or group messages are possible between two or multiple participants. One user creates a *concealed address* and shares the *read address key* and *write address key* with the other participants. The *content key* can be chosen by this user and shared, using the public keys of the other participants. Then, the participants can write messages to this *concealed address* and read the incoming messages. Whenever a new participant is added to a chat the *read address key*, *write address key*, and *content key* is shared with this user, via an encrypted message, using the public key. When a user is removed from the chat the owner of the address, knowing the *owner address key*, can change the *read address key* and *write address key*, and a new *content key* can be generated and shared with the remaining participants.

3.4 Groups

A group in an OSN is a collection of information, messages, posts, pictures, etc., comparable to a user profile. The difference is, that everything in the group may be accessed, modified, or created by multiple users. Additionally, some parts of the group may be changed by some participants. A group can be constructed like a user profile. One *concealed address* is used as a collection of different linked *concealed addresses*, containing the different information shared in the group. Because each *concealed address* can have a unique key, again a fine-grained access system can be created, where some information in the group can be accessed by all members of the group, where other information is hidden to some members. At some addresses, members of the group can add or edit information, at other

addresses, only a specified group of administrators of the group can change information. The necessary keys can be shared between the authorized users, via personal messages or shared *concealed addresses*. The construction of groups and user profiles, using *concealed addresses*, allows users to join groups, without exposing their information. Then, they just read the *concealed addresses* of this group. When users want to be visible within the group, they can publish their profile's address or their name, together with a signature, within the group.

3.5 Finding and Verifying Participants, Exchanging Keys

A user has the freedom to construct a profile, such that no other user can find him. This can be achieved when no profile information is published, or all of the user's profile fields are encrypted. Even, when all addresses on the server are tried, the corresponding *read address keys* and *content keys* have to be known. Further, if a user has encrypted profile fields, they cannot be found through the OSN. An attribute-based search for participants can be prevented. Nevertheless, when a user publishes some fields, an attribute-based search is possible. When a user does not want to be found through the OSN, the only way to find him is by receiving the needed *concealed addresses* and keys through a different channel. A channel can be via phone or by meeting in person. The user can generate a fingerprint, like a QR code or a textual representation of the *concealed addresses* and keys via a method described in [8]. This is comparable to the effort of exchanging usernames for any OSN, where the participants do not use their real names.

Public keys of other users can be verified with this method, as well: when participants A and B want to verify their keys, they call a procedure that concatenates all the public keys of A and B in a predefined order, like by ascending id of the public keys. Then a fingerprint is generated by both users A and B and compared. When both fingerprints are equal, they can be sure that the public keys are equal, and no third party has injected a wrong key. When A and B trust each other, they can use this procedure to verify the keys of other participants, as well, and construct a web of trust this way. As soon as two participants have established a trusted channel they can exchange encrypted messages through a *concealed address*. This allows them to further exchange and verify keys. When users have a verified *concealed address* to exchange messages they can work as mixes for each other. By forwarding messages to other mixes, or participants of the OSN.

When a user created a profile comparable to Fig. 2, other users can find him through the OSN, because the server can read the entry containing the name in the profile address. After all, anybody can read the *concealed address*, and the entry is not encrypted. A similar public field can be used in groups, or by mixes, as well.

3.6 Key Revocation

The revocation of keys can be achieved in multiple ways. When the *address keys* for an address are changed, a user is not allowed to access the content. Then, the *address keys* have to be redistributed to the remaining parties. Another possibility is, to move the content from the *concealed address* to another *concealed address* and carry on all communication via the new *concealed address*. Then, the new *concealed address* and the appropriate keys have to be transferred to the remaining parties. Further, the *content key* can be changed for a *concealed address*. Then, the excluded party cannot decrypt new content. Here, the *content key* has to be transferred to the remaining parties. For every method, the needed keys or *concealed addresses* can be transferred via direct messages over known *concealed addresses*.

3.7 User Registration

User registration is not needed in the proposed scheme. Every user can create a *concealed address* or multiple *concealed addresses*. This allows, that users that do not want to create a user profile to still communicate in the proposed OSN. Without registration, there is no need for authentication. Still, the server is not more vulnerable to attacks like DOS than any other server, as flooding can be prevented by the appropriate measures.

4 Privacy Analysis

The privacy of the users in the presented OSN is based on two different approaches that work together. First, all communication between users is encrypted. When using appropriate algorithms, this prevents any third person from reading messages. Here, a public-key cryptosystem can be used, where all communication uses the verified public-keys of the participants. A symmetric cryptosystem can be used, as well. Then, the keys are exchanged via a public-key cryptosystem. This approach works when multiple participants communicate. The second approach is to hide metadata, like who is communicating with whom. Our approach implements a mix network in a client-server architecture, where all participants and mixes communicate via the server. Multiple methods allow untraceable messages through a mix network: Removing of duplicate messages, such that a third person cannot trace a message and the recipient. This method is implemented by our approach, as well. Modification of messages at a mix prevents an outsider from comparing incoming and outgoing messages. This is implemented, first, when a mix decrypts the message and forwards it. Second, the mix can append any random nonce to a message, encrypt the message with the appropriate key of the recipient, and forward the modified message. Another method in a mix network is to rearrange messages, or to collect some messages and forward them at once. In our approach, this is possible, as well. A mix waits until a specific number of messages has arrived and processes them at once. Further, in our approach, a mix can join multiple messages with the same recipient.

E.g. when there are n incoming messages at a mix and m outgoing messages all three, $n = m$, $n < m$, and $n > m$, are possible. Further, in our proposed protocol, a mix can choose to forward the message to another *concealed address* of the same recipient, because participants can have access to multiple *concealed addresses*.

4.1 Attacker Model

An attacker in the OSN mainly is the provider of the OSN, or any person with access to the server and database. This can be a hacker, a disloyal administrator, or any governmental organization agent. The attacker can read all communication between the clients and the server and all entries from the database. Further, untrusted nodes in the OSN can collude with the server. Messages are not deleted by the attacker, but can be delayed. The following assumptions about the cryptographic protocols and the computational power of the attacker are made: First, calculating private-keys from public-keys for signatures and public-key cryptography is not feasible. Second, ciphertexts of the same size, generated by the same algorithm with different keys are indistinguishable. Third, ciphertexts cannot be decrypted, when the private-keys or symmetric keys are not known. Therefore, traffic analysis is prevented, because ciphertexts are indistinguishable and messages are modified at each mix.

4.2 Structure of Network (*NS*)

The leakages of the degree of a node (*NS1*) can be prevented by our approach. An attacker cannot conclude how many nodes a user knows when the user forwards all messages to the same mix or a fixed number of mixes, as long as, one mix is trusted. It is possible to obtain *concealed addresses* of participants via different channels: Users can exchange their *concealed addresses* when meeting in public. When a secured communication channel is set up, it can be used to further exchange new *concealed addresses* of other participants of a network. When the neighbors of a user are unknown, leakages about the neighbors (*NS2*) are prevented, as well.

4.3 Structure of Data (*DS*)

The structure of the stored data is hidden to any third person, as the contents of any *concealed address* are not given. Even, when an attacker can link a *concealed address* to a user, it is not possible to conclude which data is stored, when the contents are encrypted. Even a combination of different contents can be stored: it is possible to store messages, keys, and contacts in the same *concealed address*. Further, finding a connection between a *concealed address* and a user can be prevented. *concealed addresses* can be interchanged between users. When the *address keys* are exchanged, a *concealed address* can be used by multiple users, as well. When a user can read messages from a *concealed address*, it is

possible to create new *concealed addresses* with the server, using this *concealed address*. Then, the server cannot link a user with an *concealed address*. The described methods prevent third parties from concluding the amount (*DS1*), size (*DS3*), and history (*DS4*) of objects, groups, or keys, a user is in possession of. Further, no common objects (*DS2*) can be found, because contents inside *concealed addresses* can be encrypted and random nonces can be used to prevent third parties from comparing ciphertexts.

4.4 Timing (*T*)

Timing information is another leakage vector, which can be prevented by applying our scheme. Messages are collected by any mix and processed in a batch to remove any timing information. Therefore, an attacker cannot find time series patterns (*T1*). The contents of messages cannot be distinguished, which prevents to conclude about the timing between creating objects and distributing keys (*T2*) or the re-encryption of contents after a key distribution (*T3*).

4.5 Control Information (*CI*)

Hiding control information in our approach is possible, as well. IP address logging (*CI1*) and Geo-IP mapping (*CI2*) are possible but can be prevented by users when applying VPN connections or proxies. Further, when a user is sending all messages to the same trusted mix, it is still not possible to link the IP address to the owned *concealed addresses*. Together with *concealed addresses*, which can be read by multiple users, this prevents a third party from creating meaningful patterns. Analysis of access logs for objects (*CI3*) can be prevented because using *concealed addresses* allows users to establish sender privacy. Then, it is not possible to draw conclusions between a *concealed address* and a user. Further, it is not possible to make a connection between users that read from a *concealed address*. Control messages and queries for special operations, like for the login procedure (*CI4*) are not created in our approach. In fact, there are four possible actions: read from a *concealed address*, write to a *concealed address*, delete from a *concealed address*, and create a *concealed address*. This prevents third parties from analyzing other actions.

5 Performance Analysis

The proposed system heavily utilizes public-key cryptography. This can introduce long waiting-times when using the OSN. The verification of access to *concealed addresses* requires messages to be signed by the users and all nodes that are utilized in the mix. Additionally the server has to verify all signatures. To assess the feasibility, a prototype of the OSN was created. The prototype is available at: github.com/falti3/concealed. The server is mainly written in C# and provides different web API end-points. The signature-verification procedures are performed by NodeJS running JavaScript. The server-side controller

accepts messages to create *concealed addresses*. Messages can be read or deleted from and written to *concealed addresses*. For all incoming messages, the verification of the appropriate key p_R, p_W, or p_O is conducted. To store messages and addresses, a MySQL database is connected. On the client-side, a single-page HTML application with JavaScript is used. All cryptographic protocols are performed by utilizing the WebCrypto API [29]. The specific algorithms are as follows: symmetric encryption of messages (*content keys*) is performed by AES256 with CBC. Signature generation and verification (*address keys*) is performed using RSA with OAEP, modulus length 4096, and SHA256 for the key generation. The HTML application provides the basic functionalities: creating *concealed addresses*, writing messages to *concealed addresses*, and reading and deleting messages from *concealed addresses*. Mixing of messages can be simulated, by listening to addresses, forwarding the incoming messages, and answering to requests. Further, a user profile similar to that in Fig. 2 can be accessed through the client. For three message sizes of 0.5 kB, 500 kB, and 5 MB multiple random messages were created, written, read, and finally deleted again to measure the run-times of the cryptographic operations. The simulations were performed on a laptop with an Intel® Core™ i7-8550U CPU with 1.8 GHz and 16 GB of RAM. The server and MySQL database were stored on a SSD. The measurements are displayed in Table 1. The signature verification on the server was for the read and delete operations on average faster than 3.2 ms. For messages of sizes 50 kB and 5 MB, 6.81 ms and 555.12 ms were measured. This can be optimized: first, when not the whole message is verified by the server, but only a part of the message. Second, when the verification procedure is carried out in C# and not on a different NodeJS process. Still, the durations are acceptable, as most of the messages are smaller than 50 kB. On the client-side, the signature generation was faster than 4 ms for the read and delete operations and the write operations with messages of sizes 0.5 kB and 50 kB. The signature generation for 5 MB messages was 268.94 ms, which is acceptable. Again, this can be reduced, when only parts of the message are signed. For the decryption times, 1.2 ms and 1.83 ms were measured for the smaller messages. 27.66 ms were measured for 5 MB. This is acceptable and faster than the transmission time of the message between the client and the server, in most cases. When transferring messages over a mix network the durations add up, because they are performed sequentially at each node and for every message the server verifies the signature. However, three mix nodes most of the time are sufficient. When transferring files of 5 MB to the OSN still, the combined computation time for the signatures is faster than 1 s. This is acceptable. For smaller messages, that are more common, the delay generated by the cryptographic procedures most likely is not noticeable for an user.

Table 1. To evaluate the performance of the scheme, multiple write, read, and delete message operations were conducted. The following durations were measured, where "sign/encrypt" and "decryption" were measured on the client-side and the "signature verification" was measured on the server. "sign/encrypt" denotes the complete operations to encrypt a message with AES and create the signature with RSA. "signature verification" denotes the operation to verify the signature of an operation on the server. With "decryption" the operation to decrypt an encrypted ciphertext is denoted.

	Operation								
	Write			Read			Delete		
Message size	0.5 kB	50 kB	5 MB	0.5 kB	50 kB	5 MB	0.5 kB	50 kB	5 MB
Sign/encrypt (ms)	1.35	3.84	268.96	1.05	0.88	1.11	1.03	1.22	3.84
Signature verification (ms)	1.67	6.81	555.12	1.17	1.31	2.39	1.21	2.00	3.12
Decryption (ms)				1.20	1.83	27.66			

6 Conclusions

We have presented a summary of different possible privacy leakages within secure online social networks and a discussion on whether different approaches for privacy-preserving OSNs are possibly vulnerable to these leakages. Either, the OSNs are possibly vulnerable or are using a peer-to-peer architecture. Therefore, we presented a way to construct privacy-preserving, encrypted, concealed channels in a client-server architecture. No evaluable metadata is generated when using these channels, according to the previous findings of possible privacy leakages. Using these concealed communication channels different protocols are presented to provide the full functionality of OSNs. These functionalities contain a user profile, where profile fields, contact lists, pinboards, etc. can be encrypted and hidden for the server, an attacker, or any third party. Further, private messages between two or more participants of the OSN are provided. Another presented functionality are groups. A group can contain message boards, calendars, pinboards, etc. The groups can be accessed by multiple participants. Some of the participants may be only able to read content, whereas, other participants can produce content and publish it in a group. Protocols to verify public keys via secure channels are discussed, as well as the interchange of concealed addresses.

One of the main advantages of the OSN, however, can be considered as the main weak points as well: profile information or concealed channels can be hidden from the server, to prevent the server or a third party from evaluating this information. On the other hand, this prevents any user from searching this informations through the server. This means that two participants have to meet via a different channel in order to exchange the information they need in order to ultimately be able to conduct private communication via the OSN. However, this hurdle is comparable to exchanging usernames of any OSN, where the participants do not use their real name.

Acknowledgements. The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany in the framework of SoNaTe (project number 16SV7405).

References

1. Aiello, L.M., Barrat, A., Schifanella, R., Cattuto, C., Markines, B., Menczer, F.: Friendship prediction and homophily in social media. ACM Trans. Web **6**(2) (2012). https://doi.org/10.1145/2180861.2180866
2. Baden, R., Bender, A., Spring, N., Bhattacharjee, B., Starin, D.: Persona: an online social network with user-defined privacy. In: Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication, pp. 135–146 (2009)
3. Barenghi, A., Beretta, M., Di Federico, A., Pelosi, G.: Snake: an end-to-end encrypted online social network. In: 2014 IEEE International Conference on High Performance Computing and Communications, 2014 IEEE 6th International Symposium on Cyberspace Safety and Security, 2014 IEEE 11th International Conference on Embedded Software and Systems (HPCC, CSS, ICESS), pp. 763–770. IEEE (2014)
4. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2), 84–90 (1981)
5. Conti, M., Mancini, L.V., Spolaor, R., Verde, N.V.: Analyzing android encrypted network traffic to identify user actions. IEEE Trans. Inf. Forensics Secur. **11**(1), 114–125 (2016)
6. Cutillo, L.A., Molva, R., Strufe, T.: Safebook: a privacy-preserving online social network leveraging on real-life trust. IEEE Commun. Mag. **47**(12), 94–101 (2009). https://doi.org/10.1109/MCOM.2009.5350374
7. Cutillo, L.A., Molva, R., Strufe, T.: Privacy preserving social networking through decentralization. In: 2009 Sixth International Conference on Wireless On-Demand Network Systems and Services, pp. 145–152. IEEE (2009)
8. Dechand, S., Schürmann, D., Busse, K., Acar, Y., Fahl, S., Smith, M.: An empirical study of textual key-fingerprint representations. In: 25th USENIX Security Symposium (USENIX Security 2016), pp. 193–208 (2016)
9. Finney, H., Donnerhacke, L., Callas, J., Thayer, R.L., Shaw, D.: OpenPGP message format. RFC 4880, November 2007. https://doi.org/10.17487/RFC4880. https://rfc-editor.org/rfc/rfc4880.txt
10. Greschbach, B., Kreitz, G., Buchegger, S.: The devil is in the metadata—new privacy challenges in decentralised online social networks. In: 2012 IEEE International Conference on Pervasive Computing and Communications Workshops, pp. 333–339 (2012)
11. Kikuchi, H., Tada, M., Nakanishi, S.: Secure instant messaging protocol preserving confidentiality against administrator. In: 2004 18th International Conference on Advanced Information Networking and Applications, AINA 2004, vol. 2, pp. 27–30. IEEE (2004)
12. Lucas, M.M., Borisov, N.: FlyByNight: mitigating the privacy risks of social networking. In: Proceedings of the 7th ACM Workshop on Privacy in the Electronic Society, WPES 2008, pp. 1–8. ACM, New York (2008). https://doi.org/10.1145/1456403.1456405
13. Marlinspike, M.: The Double Ratchet Algorithm, November 2016. https://signal.org/docs/specifications/doubleratchet/

14. Marotta, V., Abhishek, V., Acquisti, A.: Online tracking and publishers' revenues: an empirical analysis. In: Workshop on the Economics of Information Security (2019)
15. Mcauley, J., Leskovec, J.: Discovering social circles in ego networks. ACM Trans. Knowl. Discov. Data **8**(1) (2014). https://doi.org/10.1145/2556612
16. Mislove, A., Viswanath, B., Gummadi, K.P., Druschel, P.: You are who you know: inferring user profiles in online social networks. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, WSDM 2010, pp. 251–260. Association for Computing Machinery, New York (2010). https://doi.org/10.1145/1718487.1718519
17. Moscaritolo, V., Belvin, G., Zimmermann, P.: Silent Circle Instant Messaging Protocol - Protocol Specification, December 2012. https://web.archive.org/web/20150402122917/silentcircle.com/sites/default/themes/silentcircle/assets/downloads/SCIMP_paper.pdf
18. OTR Development Team: Off-the-Record Messaging Protocol Version 3 (2012). https://otr.cypherpunks.ca/Protocol-v3-4.1.1.html
19. Perez, B., Musolesi, M., Stringhini, G.: You are your metadata: identification and obfuscation of social media users using metadata information. In: Proceedings of the International AAAI Conference on Web and Social Media, vol. 12 (2018)
20. Piotrowska, A.M., Hayes, J., Elahi, T., Meiser, S., Danezis, G.: The loopix anonymity system. In: 26th USENIX Security Symposium (USENIX Security 2017), pp. 1199–1216 (2017)
21. Robison, C., Ruoti, S., van der Horst, T.W., Seamons, K.E.: Private Facebook chat. In: 2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Conference on Social Computing, pp. 451–460. IEEE (2012)
22. Schaad, J., Ramsdell, B., Turner, S.: Secure/multipurpose internet mail extensions (S/MIME) version 4.0 message specification. RFC 8551 (2019). https://doi.org/10.17487/RFC8551. https://rfc-editor.org/rfc/rfc8551.txt
23. Schifanella, R., Barrat, A., Cattuto, C., Markines, B., Menczer, F.: Folks in folksonomies: social link prediction from shared metadata. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, WSDM 2010, pp. 271–280. Association for Computing Machinery, New York (2010). https://doi.org/10.1145/1718487.1718521
24. Schillinger, F., Schindelhauer, C.: End-to-end encryption schemes for online social networks. In: Wang, G., Feng, J., Bhuiyan, M.Z.A., Lu, R. (eds.) SpaCCS 2019. LNCS, vol. 11611, pp. 133–146. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24907-6_11
25. Singh, M.: Telegram hits 400m monthly active users, April 2020. https://techcrunch.com/2020/04/24/telegram-hits-400-million-monthly-active-users/
26. Threema: Threema Cryptography Whitepaper, January 2019. https://threema.ch/press-files/cryptography_whitepaper.pdf
27. Tyagi, N., Gilad, Y., Leung, D., Zaharia, M., Zeldovich, N.: Stadium: a distributed metadata-private messaging system. In: Proceedings of the 26th Symposium on Operating Systems Principles, pp. 423–440 (2017)
28. Van Den Hooff, J., Lazar, D., Zaharia, M., Zeldovich, N.: Vuvuzela: scalable private messaging resistant to traffic analysis. In: Proceedings of the 25th Symposium on Operating Systems Principles, pp. 137–152 (2015)

29. Web Cryptography API - W3C Recommendation 26 January 2017. https://www.w3.org/TR/2017/REC-WebCryptoAPI-20170126/
30. Yang, C.H., Kuo, T.Y., Ahn, T., Lee, C.P.: Design and implementation of a secure instant messaging service based on elliptic-curve cryptography. J. Comput. **18**(4), 31–38 (2008)