Key Management in Distributed Online Social Networks

Felix Günther Mark Manulis Thorsten Strufe
TU Darmstadt *CASED* *Uni Mannheim*
Email: {guenther,strufe}@cs.tu-darmstadt.de, mark@manulis.eu

***Abstract*—Decentralized approaches for online social networks (OSNs) have been of recent research interest, enabling users to create profiles and share data like in other OSNs as, e.g., Facebook. Since the decentralized architecture does not contain a central authority that is able perform access control, encryption is needed to ensure the confidentiality of published data. This paper outlines strict requirements and weak constraints for the encryption of data attributes in decentralized OSNs. Subsequently, an overview of possible cryptographic solutions is given and their suitability according to these requirements is analyzed. As a result, the differences and trade-offs between and within the given approaches are expounded. The outcome of this paper can be used as a foundation for further investigations on this topic.**

I. Introduction

Recently, many different decentralized approaches for online social networks (OSNs) have been proposed [1], [3] in an attempt to avoid the centralized control and omnipotent access of commercial service providers.

Due to their decentralized nature, those approaches lack a central authority enforcing access control on all user profiles which is possible on centralized OSNs like *Facebook* or *LinkedIn* that are accessed through a single web interface. In contrast to these systems, the profile of a user in decentralized OSNs is usually stored on her system itself, all user systems forming a peer-to-peer network. As the profile of a user may be replicated on the systems of her friends (i.e., her contacts) for accessibility reasons, she may not be able to enforce live access control on her profile either.

These constraints yield the need of encryption of user profile data in decentralized OSNs to guarantee its confidentiality. In addition to that, users shall be enabled to restrict access to their profiles in a fine-grained manner on atomic attributes. Thus, the encryption scheme has to offer a possibility to encrypt a multitude of single attributes each for a single user or a group of users.

Aside of these hard constraints there exist some weaker constraints. As all operations have to be executable on devices with restricted resources – as, e.g., users should be able to log in via their mobile phone – and with affected users being offline, storage space requirements and the requisite interaction between group members in the particular encryption scheme have to be taken into account.

In order to be able to evaluate the weaker constraints in a concrete setting and extract the trade-off between them, we focus the analysis of the proposed key management schemes to *Safebook*, a decentralized approach for an online social network proposed by Cutillo et al. [3]. This course of action allows for an exact examination of the different schemes – e.g., regarding the storage overhead they impose – without loosing the general applicability of the presented schemes.

After presenting architectural structures of Safebook that are relevant for key management, this paper points out the significant requirements demanded from the encryption scheme and outlines weaker constraints leading to varying focuses among the investigated approaches. Based on these requirements and constraints, an overview of different encryption schemes is given, reflecting different approaches to distribute and manage group keys. Thereupon, these schemes are analyzed regarding the weak constraints by developing abstract formulas for several interesting properties of the presented approaches which are evaluated later on using reasonable system parameters.

As a result, it is shown that the examined schemes differ heavily in terms of the given constraints, whereas an approach based on broadcast encryption performs best regarding the outlined properties.

The rest of this paper is organized as follows. First, beneficial architectural structures of Safebook are pointed out in section II. In section III, requirements for suitable encryption schemes are discussed and weaker constraints identified. A survey of applicable approaches is given thereafter in section IV, followed by an evaluation of these in section V. In section VI, approaches related to the described ones are discussed and their drawbacks regarding the given requirements are pointed out. The paper concludes in section VII with a short summary and gives an outlook on future work.

II. Beneficial Structure of Safebook

The architecture of Safebook includes a *Trusted Identification Service* (TIS) that provides each user joining the network with an unambiguous node identifier and pseudonym. Along with this, each user generates two public/private key pairs for the peer-to-peer and OSN levels for which she receives certificates from the TIS.

Furthermore, the peer-to-peer substrate of Safebook allows to resolve the public key belonging to a user or to a user's pseudonym. That way, encryption schemes can utilize the global availability of keying material (i.e., distributed public/private key pairs) for their purpose. The existence of keying material eases the communication and key distribution needed for key management and renders more complex approaches (as they are needed in, e.g., ad-hoc networks, where no preliminary keying material is available) unnecessary.

978-1-4577-0351-5/11/$26.00

Beyond that, the user groups who shall be allowed to access certain attributes are – in contrast to members in, e.g., ad-hoc networks or interest groups in Facebook – both relatively stable and more likely increasing than decreasing. Thus, member exclusions will occur only rarely, which can be advantageous for some encryption schemes.

III. Requirements and Constraints

In this section, the mandatory requirements for encryption schemes solving the given problem are defined first, followed by weaker constraints, whose degree of fulfillment can be quantified for each approach.

A. Mandatory Requirements

The following requirements have to be met by an approach to be suitable at all:

1) Confidentiality: If an attribute a is encrypted for a certain group of users $U_a = \{U_{a,1}, U_{a,2}, \ldots, U_{a,n}\}$, it has to be computationally infeasible for any user $U \notin U_a$ to decrypt the attribute a.

2) Access Control: Only the owner of a profile can change the access rules to its attributes, defining who is allowed to access a certain attribute and who is not. In particular, the mirroring peers (in case of Safebook, a peer mirrors the profiles of all her contacts for accessibility reasons) must not be able to manipulate the access rules, neither of attributes they are allowed to decrypt nor of these that they are not allowed to decrypt.

3) Privacy: It has to be infeasible for any user to discover the identity of an authorized user (except for herself) of an attribute as well as to decide whether any other user is or is not authorized to access an attribute.

4) Key Independence: If the encryption schemes use group keys $\mathcal{K} = \{K_0, K_1, \ldots, K_n\}$ (i.e., a secret share is published to and known by all users having access to a given attribute), it has to be guaranteed that a passive adversary knowing an arbitrary subset $\hat{\mathcal{K}} \subset \mathcal{K}$ of group keys is not able to discover any other group key $\bar{K} \in (\mathcal{K} \setminus \hat{\mathcal{K}})$ (cf. [11]). Key independence implies forward and backward secrecy; i.e., an attacker knowing a contiguous subset of group keys cannot discover subsequent or preceding keys.

B. Weaker Constraints

Besides the hard requirements given above, there are weaker constraints posed by the architecture of Safebook, which allow for an evaluation of different approaches applying to the requirements. These are:

1) Storage Space: The keys used by encryption schemes in the given setting are duplicated in two ways: On the one hand, the keys the owner has to store in her profile (e.g., encrypted shared keys for authorized users) are replicated on the systems of all her contacts. On the other hand, the keys needed for accessing an attribute have to be stored by the client user (regarding the access to a profile) for every attribute she has access to and that for any contact's profile in her contact list.

It is obvious that especially the client-side storage needs can become very large. Therefore, storing keys on clients should be avoided if possible, or at least used on a limited scale.

Keeping the replication of user profiles in mind, the amount of storage overhead imposed in the profile should be kept as low as possible. It should be noted that not all data stored at the owner of a profile necessarily has to be stored in the profile itself, e.g., the private key of the owner clearly has to be stored on her system but – needless to say – not in her profile, replicated on other systems. Encryption schemes may introduce similar keys or other data, which have to be stored on the owner's system only, not directly in the profile.

2) Interaction with group users: Since Safebook is based on a peer-to-peer system, its users' systems apparently are not permanently online, which makes direct communication difficult. Therefore, live interaction needed between the owner of a profile and users in an access group for a certain attribute should be reduced to a minimum. Otherwise, e.g., establishment of keys would slow down dramatically, since delayed channels would have to be used.

3) Expenditure of resources needed for computations: As Safebook clients should be able to run on mobile devices with limited computing power, access control management has to be feasible also on these clients. Thus, the computation of keys is demanded not to be too expensive regarding the resources needed.

IV. Encryption Schemes

In this section, different approaches suiting the requirements are described. First, a simple and intuitive scheme in two variants is described. Thereafter, a more complex approach is outlined based on the *One-way Function Tree* (OFT) scheme [2], [12], [14], which itself bases on the *Logical Key Hierarchy* approach (LKH) [18], [19]. The third scheme – presented in [8] – uses bilinear pairings for broadcast encryption (BE) to achieve adaptive security (the scheme is subsequently referred to as "Gentry-Waters BE").

A. Simple Shared Key

The intuitive approach to encrypt attributes for a group of users is the following: The profile owner creates a new attribute a and defines the group $U_a = \{U_{a,1}, U_{a,2}, \ldots, U_{a,n}\}$ of users authorized to access it. She then chooses a secret key K_a for this attribute at random, encrypts the attribute a as $Enc_{K_a}(a)$ and adds the encrypted attribute to her profile. Finally, the key K_a has to be distributed to all users in U_a. Regarding the architecture of Safebook, there are two possibilities to distribute the key K_a, forming the two shapes of this approach:

1) Client-side Key Storage: The first variant is to send every user $U_{a,i}$ the secret key K_a for the new attribute using the respective public key pk_i for encryption; i.e., send $Enc_{pk_i}(K_a)$ to each $U_{a,i} \in U_a$.

In this case, the owner of the profile only has to store the current attribute key K_a (which does not have to – and should not – be stored in the profile), but this key needs to be stored also on the system of every user in U_a.

When creating an attribute, K_a has to be sent to each user $U_{a,i}$, resulting in n messages. If a new user $U_{a,n+1}$ is added to the group of authorized users U_a, the attribute key K_a needs to be changed and the new key K'_a has to be transmitted to all users $U_{a,i} \in U'_a = U_a \cup \{U_{a,n+1}\}$ as $Enc_{pk_i}(K'_a)$, thus resulting in $n+1$ messages and encryptions. The owner of the profile then encrypts the attribute with the new key as $Enc_{K'_a}(a)$ replacing the old encryption in the profile. On exclusion of a user $U_{a,j}$ out of U_a, the owner of the profile also has to choose a new secret key K'_a and replace the encrypted attribute in the profile. The key has to be distributed to all users in the new group of authorized users $U'_a = U_a \setminus \{U_{a,j}\}$, resulting in $n-1$ messages and encryptions. It should be noted that this approach also supports the addition or exclusion of a subgroup of multiple users $\bar{U}_a = \{\bar{U}_{a,j_1}, \ldots, \bar{U}_{a,j_m}\}$ at once: Addition and exclusion of this group can be carried out like the addition or exclusion of a single user, publishing the new key K'_a to $U'_a = U_a \cup \bar{U}_a$ in case of user addition respectively $U'_a = U_a \backslash \bar{U}_a$ in case of user exclusion, resulting in $n+m$ respectively $n-m$ messages.

2) Profile-side Key Storage: The second variant of this approach is to store the secret key in the profile rather than distributing it to all authorized users. For this purpose, the owner of the profile computes $Enc_{pk_i}(K_a)$ for each $U_{a,i}$; i.e., she encrypts the secret key K_a for each authorized user $U_{a,i}$ using the respective public key pk_i. These encodings of K_a are then stored in the owners profile, accessible for all Safebook users, thus also the authorized ones.

This way, the authorized users do not need to store anything: To access a an authorized user $U_{a,i}$ decrypts the encryption of K_a destined for him using his private key sk_i and receives K_a enabling him to decrypt the attribute. This zero-storage at client-side is traded in for greater storage needs at profile-side, since the owner of the profile in this variant has to store not only K_a (outside the profile), but also n encryptions of K_a in the profile that are replicated on the systems of her contacts.

Since no information has to be transferred to the authorized users in this variant, there is no group interaction at all; i.e., no messages need to be sent. Member addition and exclusion (also of multiple users) are done analogously to the first variant, storing the new encrypted keys in the profile rather than sending them to the users.

B. OFT-based Approach

The approach based on the One-way Function Tree (OFT) uses a binary tree, containing the shared secret key K_a for the attribute a at its root and associating the leafs with the n authorized users $U_{a,1}, \ldots, U_{a,n}$ (see Figure 1).

The key tree is of height $\log n$ and is initialized as follows (cf. [2]): The profile owner associates every node v with a randomly chosen key $K_{a,v}$ and sends each user all keys associated to nodes on the path from the user to the root encrypted with the respective user's public key. In the tree of Figure 1 for example, $U_{a,1}$ would receive $K_{a,00}$, $K_{a,0}$ and K_a. Thus, each user receives at most $\log n + 1$ keys, transmitted with n messages. As all users know K_a, the encrypted attribute $Enc_{K_a}(a)$ can be stored in the profile with all authorized users able to decrypt it.

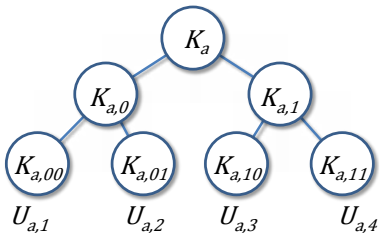


Figure 1. Exemplary OFT key tree

On user removal, all keys associated to nodes on the path from the removed user $\bar{U}$ to the root have to be changed to assure forward secrecy. As an enhancement of the LKH approach, OFT does not choose all new keys on the path to the root at random, but only assigns the parent node $p(\bar{U})$ of the removed user $\bar{U}$ a randomly chosen value r. Then, a pseudo-random generator [9] G which doubles the size of its input ($L(x)$ and $R(x)$ denoting the left and right halves of the output of $G(x)$) is used to determine the new keys on the path to the root. Every other node v on the path to the root is assigned a value r_v computed as $r_{p(v)} = R(r_v) = R^{|\bar{U}|-|v|}(r)$ (where $p(v)$ denotes the parent and $|v|$ the height of v). Based on these values the new key of a node v is defined as $K'_{a,v} = L(r_v) = L(R^{|\bar{U}|-|v|-1}(r))$. Finally, each value $r_{p(v)}$ is encrypted with the key $K_{a,s(v)}$ ($s(v)$ denoting the sibling of v) and sent to the users in the subtree of $s(v)$, thus enabling all users to compute the new attribute key K'_a. For example, if user $U_{a,2}$ is removed in the tree of figure 2, $Enc_{K_{a,011}}(r)$ has to be sent to $U_{a,3}$, $Enc_{K_{a,00}}(R(r))$ to $U_{a,1}$ and $Enc_{K_{a,1}}(R(R(r)))$ to $U_{a,4}$ and $U_{a,5}$, thus $\lceil \log n \rceil$ encryptions are needed and $n-1$ messages sent. Now, each user is able to compute the new keys $K'_{a,01} = L(r)$, $K'_{a,0} = L(R(r))$ and $K'_a = L(R(R(r)))$.

User addition is accomplished similar to user removal. To guarantee backward secrecy, all keys on the path from the new user to the root have to be changed the same way as if the new user would have been removed. Thus, the addition of $U_{a,2}$ to the tree of figure 2 results in the same encryptions

and $n+1$ messages sent (of course, r is chosen newly at each addition or removal). If user $U_{a,3}$ is moved down in the tree in order to add $U_{a,2}$, she keeps her old key $K_{a,01}$ as the new key $K_{a,011}$.

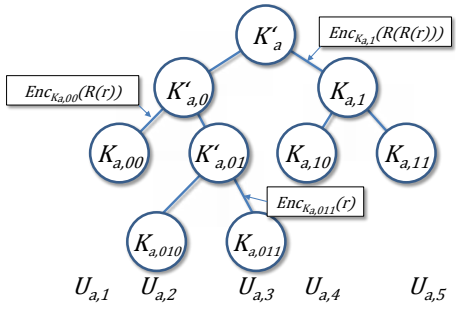


Figure 2. OFT user removal or addition

The owner of the profile has to store the whole key tree (*not* in the profile itself), whereas the authorized users have to retain at most $\lceil \log n \rceil + 1$ keys each.

C. Gentry-Waters broadcast encryption (BE) approach

The Gentry-Waters BE scheme presented in [8], which is a very novel approach in the field of broadcast encryption [4], [5], [7], [13] (especially regarding adaptive security), can be used for the encryption of multiple attributes at the same time and with low overhead in our setting. Due to its complexity, we will only sketch the approach at this point (cf. [8], section 3.1 for more details). The construction contains the four algorithms *Setup*, *KeyGen*, *Enc* and *Dec* which we will draft subsequently:

Setup generates the basis groups $\mathbb{G}, \mathbb{G}_T$ of prime order p and the bilinear map e. Moreover, it chooses $\alpha \in \mathbb{Z}_p$ and $g, h_1, \ldots, h_n \in \mathbb{G}$ at random and computes a public and a private key PK and SK. Using the private key, *KeyGen* is called for each of n users (where n constitutes the upper bound for the number of users which can be granted access to an attribute), resulting in a secret key d_i for each user. These secret keys can be stored in the profile, encrypted with the public key of the respective user. This concludes the initialization.

Thereafter, a secret share for each group of authorized users can be computed by providing SK and the group of authorized users to *Enc*, which outputs a header Hdr and the secret key K. Using this secret key, the owner of the profile can now encrypt the attribute, the group shall have access to and store it in the profile together with the header Hdr. Each authorized user is then able to decrypt the shared key K using *Dec* with her secret key d_i[1], which she is able to decrypt with her private key sk_i.

User addition and removal requires a new execution of *Enc*, since the group of authorized users has changed. A regeneration of the secret keys d_i is not needed.

The authorized users have to store nothing in this approach. The owner of the profile has to retain the private key SK whereas the public key PK and the encrypted secret keys d_i for each user as well as the header Hdr and symmetric encryption $Enc_K(a)$ for each attribute have to be stored in the profile.

V. Evaluation

We will now evaluate the encryption schemes described in the previous section according to the requirements and weaker constraints presented in section III. First, the relevant metrics (as storage space, numbers of encryptions needed, etc.) for analysis are defined. Then, abstract formulas are determined for all properties and encryption schemes. In a third step, we apply concrete values for the parameters of the properties to explore the trade-offs imposed by the approaches. Finally, the differences between the approaches are discussed.

A. Property definitions

We will study the following abstract properties on each scheme:

1) Storage requirements at the profile owner outside the profile: The amount of storage in bytes needed for a single attribute on the system of the profile owner, which is *not* stored in the profile, is denoted by S_o.

2) Storage requirements in the profile and its replications: The amount of storage in bytes needed for a single attribute in the profile is denoted by S_p. This value includes the storage needed on the systems of the profile owner's contacts (remember that Safebook profiles are replicated at their owner's contacts for accessibility reasons). It does not include the storage needed for the symmetric encryption of the attribute itself.

3) Storage requirements at the authorized users: The amount of storage in bytes needed for a single attribute on the systems of *all* users authorized to access the attribute is denoted by S_u.

4) Number of encryptions on initialization: The number of encryptions needed on initialization of the encryption scheme is denoted by E_i, not including the encryption of the attribute itself.

5) Number of encryptions on user addition: The number of encryptions needed when a user is added to the group of authorized users is denoted by E_a, not including the encryption of the attribute itself.

6) Number of encryptions on user removal: The number of encryptions needed if a user is excluded from the group of authorized users is denoted by E_r, not including the encryption of the attribute itself.

[1]The *Dec* algorithm also has to be provided with the indices of the users that are allowed to access a certain attribute. It is arguable how much information about the users with the respective indices can be deduced. We assume that meaningful linking becomes (statistically) impossible if the attribute is encrypted for a certain percentage of additional dummy indices not related to any user.

Table I
ABSTRACT PROPERTY FORMULAS

Scheme	Storage			Encryptions			Messages		
	S_o	S_p	S_u	E_i	E_a	E_r	M_i	M_a	M_r
S. Shared Key (1)	b_s	0	Nb_s	N	$N+1$	$N-1$	N	$N+1$	$N-1$
S. Shared Key (2)	b_s	$(C+1)Nb_a$	0	N	$N+1$	$N-1$	0	0	0
OFT	$(2N-1)b_s$	0	$N(\lceil \log N \rceil + 1)b_s$	N	$\lceil \log N \rceil$	$\lceil \log N \rceil$	N	$N+1$	$N-1$
Gentry-Waters BE	$\frac{b_g}{A}$	$\left(\frac{\lvert PK \rvert + Cb_a}{A} + 2b_g\right)(C+1)$	0	1	1	1	0	0	0

7) Number of messages on initialization: The overall number of messages sent on initialization of the encryption scheme is denoted by M_i.

8) Number of messages on user addition: The overall number of messages sent when a user is added to the group of authorized users is denoted by M_a.

9) Number of messages on user removal: The overall number of messages sent if a user is excluded from the group of authorized users is denoted by M_r.

B. Abstract property formulas

Table I shows the abstract formulas for all properties and encryption schemes using the notation shown in table II. We will now develop these formulas:

1) Simple Shared Key: In both variants the owner has to store the shared symmetric key K_a, thus $S_o = b_s$. $S_p = 0$ for the variant of client-side key storage (since the profile is not needed for storage here) and $S_p = (C+1) \cdot N \cdot b_a$ for the profile-side storage, as we have to store the (asymmetrically) encrypted K_a for N users (N is the number of users authorized to access attribute a, cf. table II) and this storage is replicated to the C contacts of the profile's owner. Each authorized user has to store K_a in the client-side variant, thus $S_u = N \cdot b_s$ here. In the profile-side variant, the users have to store nothing.

The number of needed encryptions is identical for both variants: On initialization, the shared key K_a has to be encrypted for each user, resulting in N encryptions. When adding or removing a user, K_a has to be encrypted for all members of the new group of authorized users, thus $N+1$ respectively $N-1$ encryptions have to be performed.

Whilst the profile-side variant does not need any messages to be sent, each encrypted key has to be transmitted to the appropriate user in the client-side variant.

2) OFT-based Approach: In the OFT-based scheme, the owner of the profile has store the key tree outside of the repository which contains $2N-1$ nodes, each associated with a symmetric key b_s. The N users in the tree have to retain the keys on the path to the root (at most $\lceil \log N \rceil + 1$ symmetric keys of size b_s), thus $S_u = N \cdot (\lceil \log N \rceil + 1) \cdot b_s$. Nothing has to be stored in the profile.

During the initialization, all keys on the path from a user's leaf node to the root have to be sent encrypted to the respective user, resulting in N encryptions and N messages sent. Since user addition and removal are quite similar, they both need the same number of encryptions: For each subtree under a sibling of a node on the path from the removed or added node to the root, a value r_v is encrypted, resulting in at most $\lceil \log N \rceil$ (and at least $\log N$) encryptions. All nodes in the tree have an ancestor changing its associated key, thus all $N+1$ respectivley $N-1$ nodes have to receive a key updating message.

Table II
NOTATION USED IN THE PROPERTY FORMULAS

A	the number of attributes in the profile
N	the number of users authorized to access an attribute
C	the number of contacts of the profile's owner
n_{BE}	the maximum number of users chosen for the *Setup* algorithm
b_s	the size of a symmetric key in bytes
b_a	the size of an asymmetric key in bytes
b_g	the size of a pairing-based group element ($g \in \mathbb{G}$) in bytes
b_p	the size of a pairing ($e(g,g)$) in bytes

3) Gentry-Waters BE scheme: The owner of the profile has to store the private key SK of size b_g in the Gentry-Waters BE scheme. As this key has to be stored only once, its size has to be distributed over the number of attributes; i.e., divided by the number of attributes A, resulting in $S_o = \frac{b_g}{A}$. The authorized users need not retain anything. The profile has to provide the public key PK (consisting of $n_{BE}+1$ elements of the group $\mathbb{G}$ and one pairing $e(g,g)^\alpha$) of size $|PK| = (n_{BE}+1) \cdot b_g + b_p$ and the encrypted secret keys d_i for all of the profile user's contacts ($C \cdot b_a$). This storage – like the private key – is required only once and therefore divided by A. Further, the header Hdr of size $2 \cdot b_g$ has to be stored in the profile for each attribute. Finally, all this is replicated on the C systems of the user's contacts. Summarized, $S_p = \left(\frac{|PK| + C \cdot b_a}{A} + 2 \cdot b_g\right) \cdot (C+1)$.

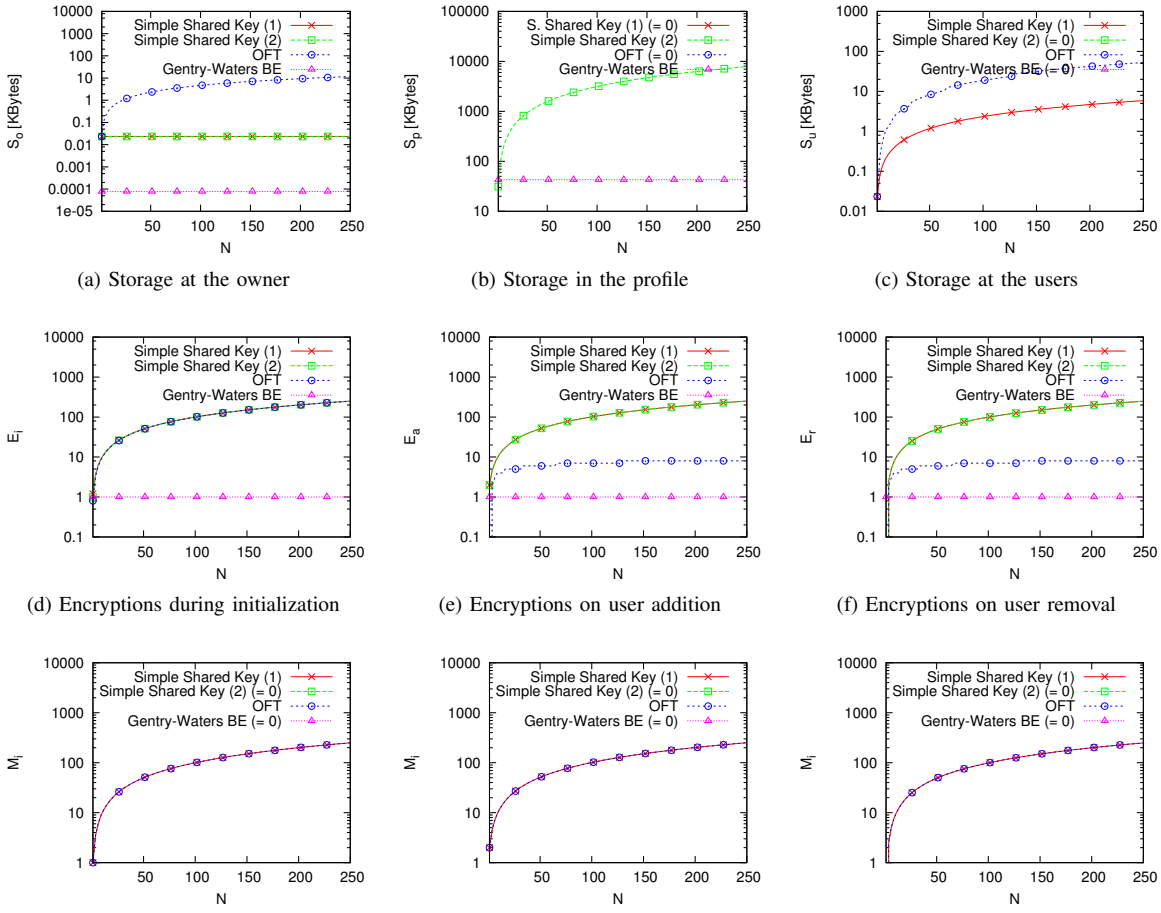
Concerning the number of encryptions, we only consider needed executions of the algorithm *Enc*, since *Setup* and *KeyGen* have to be processed only at the initialization of the scheme, not at each initialization of an attribute. Thus, we have a single encryption for initialization as well as for user addition and removal.

No active messaging is needed in this approach.

C. Analysis of the proposed schemes

Figure 3 shows the plots of all properties over the number of authorized users for an attribute.

For A, C, n_{BE}, b_a, b_s, b_g and b_p, we have used the following reasonable values: We have assumed a high average of $C = 250$ contacts and a medium average of $A = 300$ attributes in a profile. A maximum of 250 users for a group of authorized users used in the *Setup* algorithm of the BE approach seems reasonable. Furthermore, we have chosen a



(a) Storage at the owner (b) Storage in the profile (c) Storage at the users

(d) Encryptions during initialization (e) Encryptions on user addition (f) Encryptions on user removal

(g) Messages during initialization (h) Messages on user addition (i) Messages on user removal

Figure 3. Plots of all properties over the number of authorized users N (logscale, properties of constant 0 are marked accordingly)

bit length of 1024 for asymmetric keys and pairings each ($b_a = 128$, $b_p = 128$) and a bit length of 192 for symmetric keys and pairing-based group elements each ($b_s = 24$, $b_g = 192$).

D. Discussion

The two variants of the *Simple Shared Key* approach require a considerable amount of storage either in the profile or at client-side, which is rather problematic since storage in the range of megabytes at profile-side respectively kilobytes at client-side is needed just for a *single* attribute. Keeping in mind that a user may have thousands of attributes and clients have to store attribute keys for each attribute of all of their contacts, the *Simple Shared Key* approach gets infeasible rapidly.

Without doubt, the great advantage of the approach based on the One-way Function Tree scheme is the ease of access revocation; i.e., user removal in our case. However, this scheme requires a comparatively high amount of group interaction in form of messages to the authorized users and depends on an amount of client-side key storage that is even higher than the one needed by the Simple Shared Key approach. As stated before, the amount of client-side storage is multiplied by the number of attributes the user has access to at the profiles of all her contacts. Thus, this approach results in large overall storage needs.

It is obvious that the Gentry-Waters BE approach based is most suitable regarding the given constraints. Especially the important storage requirements are met as the Gentry-Waters BE scheme performs very well at each of the three related properties. Even if it is considered that pairing based cryptography is more expensive than symmetric or public key cryptography, the Gentry-Waters BE approach requires a tolerable amount of computing power. Moreover, the avoidance of group interaction in form of messages is an advantage of this scheme.

Given the presented evaluation, the Gentry-Waters BE approach turns out to be the best fitting approach, worthy of further investigation.

VI. Related Approaches

Steiner, Tsudik and Waidner proposed an approach that extends Diffie-Hellman key exchange to groups in [15], [16]. This scheme provides cheap member addition but is based on the contribution of all group members to compute the group key, which poses two problems regarding the requirements and constraints: It requires direct interaction between the authorized users, which is infeasible regarding the peer-to-peer architecture. Moreover, the key exchange happens directly between the members of the authorized user group,

which violates the privacy requirements as the authorized users have to know each other.

A recent encryption scheme proposed by Eskeland and Oleshchuk [6] uses fractional public keys to compute a shared group key. If Safebook users could be provided with a private key based on this scheme, neither they nor the owner of the profile would have to store any further keys. However, the private keys are generated by a central trusted authority like Safebook's Trusted Information Service, which would then be able to decrypt any communication in Safebook. The authorized users also need to know the other authorized users, conflicting with the privacy requirements.

In [10], Jin and Lotspiech present a broadcast encryption scheme with differently privileged users. They introduce "security classes" to provide different data sets for differently privileged user groups. Even though Safebook attribute user groups could be arranged in a hierarchy like in role-based systems, the proposed approach – since it is designed for a linear hierarchy (i.e., group A > group B > group C, etc.) – imposes conflicts regarding overlapping attribute user groups and key distribution.

Multiple approaches exist to establish group keys in (mobile) ad hoc networks (cf. [17]) that could generally be used to set up attribute group keys in Safebook. However, these approaches assume that no preliminary keying material is available and therefore impose much useless overhead in a system like Safebook having public/private key pairs at hand for bilateral communication.

VII. Conclusion and Future Work

The architecture of decentralized online social networks leads to the need for different encryption schemes as they are used in centralized networks. Private data has to be encrypted in user profiles to guarantee its confidentiality.

The peer-to-peer structure of those networks raises different requirements and constraints that we have presented in this paper, subdivided into strict and weaker ones. Afterwards, different encryption schemes have been described that base upon distinct approaches.

To evaluate the given schemes, interesting properties stemming from the outlined requirements and constraints are defined first. As a second step, abstract formulas for the computation of these properties are developed for each encryption scheme using varying system parameters. All properties are plotted afterwards with reasonable values applied for the abstract parameters, constituting a basic overview over the trade-offs imposed between and within different approaches. For the sake of concrete results and without loss of general applicability, the evaluation of the proposed schemes has been performed in the setting of a concrete distributed online social network, namely Safebook.

Though there is evidence that the broadcast encryption based approach performs best in the present setting, further investigations and simulations – especially simulations based upon real data sets – are needed to provide more insight into the characteristics of the presented approaches. These as well as an elaborate security analysis of the proposed schemes from the cryptographic point of view remain for future work.

References

[1] S. Buchegger, D. Schiöberg, L.-H. Vu, and A. Datta. PeerSoN: P2P social networking: early experiences and insights. In *EuroSys/SNS*, 2009.

[2] R. Canetti, J. A. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. In *INFOCOM*, 1999.

[3] L. A. Cutillo, R. Molva, and T. Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *"IEEE Communications Magazine"*, 2009.

[4] Y. Dodis and N. Fazio. Public key broadcast encryption for stateless receivers. In *Digital Rights Management Workshop*, 2002.

[5] Y. Dodis and N. Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In *Public Key Cryptography*, 2003.

[6] S. Eskeland and V. A. Oleshchuk. Secure group communication using fractional public keys. In *ARES*, 2010.

[7] A. Fiat and M. Naor. Broadcast encryption. In *CRYPTO*, 1993.

[8] C. Gentry and B. Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In *EUROCRYPT*, 2009.

[9] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 1986.

[10] H. Jin and J. Lotspiech. Broadcast encryption for differently privileged. In *SEC*, 2009.

[11] Y. Kim, A. Perrig, and G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. In *ACM CCS*, 2000.

[12] D. A. McGrew and A. T. Sherman. Key establishment in large dynamic groups using one-way function trees. Technical report, TIS Labs at Network Associates, Inc., 1998.

[13] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO*, 2001.

[14] A. T. Sherman and D. A. McGrew. Key establishment in large dynamic groups using one-way function trees. *IEEE Trans. Software Eng.*, 2003.

[15] M. Steiner, G. Tsudik, and M. Waidner. Diffie-hellman key distribution extended to group communication. In *ACM CCS*, 1996.

[16] M. Steiner, G. Tsudik, and M. Waidner. Cliques: A new approach to group key agreement. In *ICDCS*, 1998.

[17] J. van der Merwe, D. S. Dawoud, and S. McDonald. A survey on peer-to-peer key management for mobile ad hoc networks. *ACM Comput. Surv.*, 2007.

[18] D. Wallner, E. Harder, and R. Agee. Key management for multicast: Issues and architectures. RFC 2627 (Informational), 1999.

[19] C. K. Wong, M. G. Gouda, and S. S. Lam. Secure group communications using key graphs. In *SIGCOMM*, 1998.