Cryptographic Treatment of Private User Profiles

Felix Günther, Mark Manulis, and Thorsten Strufe

TU Darmstadt & CASED, Germany
{guenther, strufe}@cs.tu-darmstadt.de,mark@manulis.eu

Abstract. The publication of private data in user profiles in a both secure and private way is a rising problem and of special interest in, e.g., online social networks that become more and more popular. Current approaches, especially for decentralized networks, often do not address this issue or impose large storage overhead. In this paper, we present a cryptographic approach to *private profile management* that is seen as a building block for applications in which users maintain their own profiles, publish and retrieve data, and authorize other users to access different portions of data in their profiles. In this course, we provide: (i) formalization of *confidentiality* and *unlinkability* as two main security and privacy goals for the data which is kept in profiles and users who are authorized to retrieve this data, and (ii) specification, analysis, and comparison of two private profile management schemes based on different encryption techniques.

1 Introduction

Publishing personal profiles and other means of sharing private data are increasingly popular on the web. Online social networks (OSN) arguably are the most accepted networked service, today. Facebook alone, serving a claimed base of over 500 Million active users[1], surpassed google, and currently enjoys the highest utilization duration by their users and one of the highest access frequencies of all web sites since January 2010[2]. Its users share 90 pieces of content per month on average, mainly consisting of personally identifiable information. Protecting this data against unauthorized access is of utmost importance, since users store private and sensitive data in their OSN profiles.

The *confidentiality* of published data, meant to be shared with only a chosen group of users, is already important in centralized services. Yet, it becomes even more pressing when establishing decentralized OSN, which have been proposed recently [8,4,13] in an attempt to avoid the centralized control and omnipotent access of commercial service providers. *Unlinkability* is the more subtle requirement of protecting the identity of users who are successively interact or access certain chunks of published data. It frequently is missed and only few general solutions achieve this privacy goal [2].

A serious corpus of solutions has been proposed to address these issues in the past. Yet, there so far exist no appropriate definitions for secure and private management of user profiles, even from the cryptographic point of view, as we show in Section 2.

In this paper, we hence take a cryptographic approach to address the management of user profiles in a secure and privacy-friendly way. To this end, our goal is to come up

[1] http://www.facebook.com/press/info.php?statistics, Oct 2010
[2] http://blog.nielsen.com/nielsenwire/, Oct 2010

with a well-defined model and provably secure solutions, both to ensure the confidentiality of data, which individually is published by users in their social profiles, as well as for the privacy of the users who are allowed to access this data. For this purpose we formally define confidentiality and privacy in the given context, first. Our model further addresses several fundamental properties of a profile management scheme (PMS). They comprise of the ability of users (profile owners) to publish and remove data, as well as their ability to grant, modify, and revoke access rights to the published data. In particular, we consider PMS as an independent building block, without relying on the higher-level application or any other parties to perform these tasks.

After the specification of the model, we describe two provably secure solutions that use different techniques: our first solution, called PMS-SK, combines symmetric encryption with shared keys that are then distributed amongst the authorized users. Our second solution, called PMS-BE, involves broadcast encryption techniques. Both solutions have their advantages and disadvantages with respect to their performance and privacy, as we show in our subsequent analysis. In particular, PMS-SK provides confidentiality and perfect unlinkability, but imposes an overhead of keys linear in the number of attributes a user is allowed to access, and which have to be stored by her. PMS-BE reduces the key overhead to a constant value at the cost of lower privacy, expressed through the requirement of *anonymity*, which we also model and formally relate to the stronger notion of unlinkability. We further discuss the trade-off between privacy and efficiency by evaluating some complexity characteristics of both approaches and suggest several optimizations that could enhance their performance, while preserving their security and privacy guarantees.

The rest of the paper is organized as follows: In Section 2 we discuss previous cryptographic and non-cryptographic work on private management of user profiles. In Section 3, we introduce our formal model for such schemes and define two requirements: confidentiality and unlinkability. In Sections 4 and 5 we specify our PMS-SK and PMS-BE solutions, evaluate their complexity (including possible optimizations) and formally address their security and privacy properties as well as the notion of anonymity. In Section 6 we discuss the impact of both schemes on real-world online communities such as Facebook, Twitter and Flickr.

2 Related Work

Substantial amount of work has been done in the field of secure and private publication of sensitive data in online social networks (OSNs), demonstrating threats and proposing countermeasures. For example, Gross et al. [14] and Zheleva et al. [24] studied how access patterns of users to the information stored in user profiles and how membership of users in different groups can be exploited for the disclosure of private data. Amongst the non-cryptographic solutions is the approach proposed by Carminati et al. [7], where access to private data is modeled using semantic rules taking into account the depth of social relationships and the amount of trust amongst the users. In addition to being semi-centralized, this approach requires synchronous communication — a significant limitation in our case. There exist several cryptographic approaches to improve confidentiality and privacy in existing, mostly centralized OSNs: Lucas et. al [16] presented

flyByNight, an application to encrypt sensitive data in Facebook. Tootoonchian [21] proposed a system called *Lockr* to improve privacy in both centralized and decentralized social networks. Yet, both approaches are not able to keep security and/or privacy up under certain attacks: *flyByNight* relies on the Facebook servers as middlemen and thus enables them to introduce malicious code or keys whilst in *Lockr* malicious users are able to reveal relationship keys or disclose relationship metadata for access control, compromising privacy properties of the system. Another cryptographic approach is *Scramble!* [19], a Firefox plugin that uses the OpenPGP standard [5] to encrypt data relying on its public-key infrastructure. Moreover, Scramble! tries to achieve recipient anonymity by omitting the public identifiers of recipients in the ciphertext and allows for data storage on third-party storage using "tiny URLs", thus reducing the size of ciphertexts. Nevertheless, the approach implies linear storage overhead and, as it relies on OpenPGP, is vulnerable to active attacks as shown by Barth et al. [2]. A number of solutions aim at binding the access to private data with some fine-grained access policies. For example, Graffi et al. [12] implemented an approach based on symmetric encryption of profile items with independent shared keys, yet without actually specifying or formally analyzing the desired security and privacy properties. OSN *Persona*, presented by Baden et al. [1], implements ciphertext-policy attribute-based encryption (CP-ABE) [3] for the enforcement of access rules to the encrypted profile data (e.g., " 'neighbor' AND 'football fan' "). Their approach aims at confidentiality of attributes but does not guarantee privacy. Similarly, *EASiER* [15] which in addition to CP-ABE requires a semi-trusted proxy for the update of decryption keys upon revocation events does not consider privacy. Recently, Zhu et al. [25] proposed a collaborative framework to enforce access control in OSNs by the use of a new group-oriented convergence cryptosystem. Their scheme is centralized and focuses on joint publication of data within communities, less on the individual users and protection of their own profiles and data. A somewhat more general construct for privacy-preserving distribution of encrypted content was proposed by Barth et al. [2] using public-key broadcast encryption. Of particular interest is their notion of *recipient privacy*, which is supposed to hide the identities of recipients of the broadcast content and can be applied for the private distribution of shared keys in our PMS-SK approach (cf. Remark 1) at the cost of linear storage overhead in the number of recipients. Their scheme could also be used as a building block for a private profile management scheme that would, however, require linear storage overhead for the distribution of ciphertexts.

3 Private User Profiles: Model and Definitions

3.1 Management of User Profiles

Users. Let $\mathcal{U}$ denote a set of at most N *users*. We do not distinguish between users and their identities but assume that each identity $U \in \mathcal{U}$ is unique. Furthermore, we assume that users can create authentic and, if necessary, confidential communication channels. This assumption is motivated by the fact that the profile management scheme will likely be deployed as a building block within an application, like an online social network, where users typically have other means of authentication. In this way we can focus on

the core functionality of the profile management scheme, namely the management of and access to the profile data.

Profiles. A *profile* P is modeled as a set of pairs $(a, \bar{d}) \in \mathcal{I} \times \{0,1\}^*$ where $\mathcal{I} \subseteq \{0,1\}^*$ is the set of possible *attribute indices* a and $\bar{d}$ are corresponding *values* stored in P. We assume that within a profile P attribute indices are unique. Furthermore, we assume that each profile P is publicly accessible but is distributed in an authentic manner by its *owner* $U_P \in \mathcal{U}$. Also, every user U owns at most one profile and the profile owned by U is denoted P_U. The authenticity of profiles means that their content can only be manipulated by their respective owner who is in possession of the corresponding profile management key pmk. Since one of the goals will be to ensure confidentiality of attributes we assume that for each publicly accessible value $\bar{d}$ there exists the actual *attribute* d and that for any pair $(a, \bar{d}) \in P$ the profile owner U_P can implicitly retrieve the corresponding d as well as the group $\mathcal{G} \subseteq \mathcal{U}$ of users who are currently authorized to access d. By $\mathcal{G}^*_{a,P}$ we denote the set of users that have ever been authorized to access the attribute indexed by a within the profile P (we assume that $U_P \in \mathcal{G}^*_{a,P}$ for all attributes in P).

We are now ready to define the syntax of the profile management scheme.

Definition 1 (Profile Management Scheme). *A* profile management scheme PMS *consists of the five algorithms* `Init`*,* `Publish`*,* `Retrieve`*,* `Delete` *and* `ModifyAccess` *defined as follows:*

`Init`(κ) : *On input the security parameter κ, this probabilistic algorithm initializes the scheme and outputs an empty profile P together with the private profile management key pmk.* `Init` *is executed by the owner U_P.*

`Publish`$(pmk, P, (a, d), \mathcal{G})$: *On input a profile management key pmk, a profile P, a pair $(a, d) \in \mathcal{I} \times \{0,1\}^*$ (such that a is not yet in the profile), and a group of users $\mathcal{G}$, this probabilistic algorithm transforms the attribute d into value $\bar{d}$, adds $(a, \bar{d})$ to P, and $\mathcal{G}$ to $\mathcal{G}^*_{a,P}$. It outputs the modified P and a retrieval key rk_U for each $U \in \mathcal{G}$ (that may be newly generated or modified). Optionally, it updates pmk.* `Publish` *is executed by the owner U_P.*

`Retrieve`(rk_U, P, a) : *On input a retrieval key rk_U, a profile P, and an attribute index a, this deterministic algorithm checks whether $(a, \bar{d}) \in P$, and either outputs d or rejects with $\perp$.* `Retrieve` *can be executed by any user $U \in \mathcal{U}$ being in possession of the key for a in rk_U.*

`Delete`(pmk, P, a) : *On input a profile management key pmk, a profile P, and an attribute index a, this possibly probabilistic algorithm checks whether $(a, \bar{d}) \in P$, and if so outputs modified profile $P = P \setminus (a, \bar{d})$. Optionally, it updates pmk and rk_U of all $U \in \mathcal{G}$ where $\mathcal{G}$ denotes the set of users authorized to access the pair with index a at the end of the execution.* `Delete` *is executed by the owner U_P.*

`ModifyAccess`(pmk, P, a, U) : *On input a profile management key pmk, a profile P, an attribute index a, and some user $U \in \mathcal{U}$ this probabilistic algorithm checks whether $(a, \bar{d}) \in P$ for some $\bar{d}$, and if so finds the set $\mathcal{G}$ of users that are authorized to access the attribute d. The algorithm then proceeds according to the one of the following two cases:*

- *If $U \in \mathcal{G}$ then it updates $\mathcal{G} = \mathcal{G} \setminus \{U\}$ (i.e., user U is removed from $\mathcal{G}$).*
- *If $U \notin \mathcal{G}$ then it updates $\mathcal{G} = \mathcal{G} \cup \{U\}$ and $\mathcal{G}^*_{a,P} = \mathcal{G}^*_{a,P} \cup \{U\}$ (i.e., U is added to both $\mathcal{G}$ and $\mathcal{G}^*_{a,P}$).*

Finally, the algorithm outputs the modified profile P. Optionally, it updates pmk and the retrieval keys rk_U of all $U \in \mathcal{G} \cup \{U\}$. `ModifyAccess` *is executed by the owner U_P.*

We remark that some profile management schemes may include an additional algorithm `ModifyAttribute` allowing U_P to modify the attribute d behind some pair $(a, \bar{d}) \in P$ in a more efficient way than by consecutive execution of `Delete` and `Publish`. However, for the security treatment of profile management schemes it is sufficient to consider only `Delete` and `Publish` that make the modification of attributes possible.

3.2 Adversarial Model

In order to define security and privacy of a profile management scheme PMS we consider a PPT adversary $\mathcal{A}$ that knows all users in the system, i.e., the set $\mathcal{U}$ is assumed to be public, and interacts with them via the following set of queries:

$\mathsf{Corrupt}(U)$: This corruption query gives $\mathcal{A}$ all secret keys known to U, including the profile management key pmk and all retrieval keys rk_U (with which U can access other users' profiles).
U is added to the set of corrupted users that we denote by $\mathcal{C} \subseteq \mathcal{U}$.

$\mathsf{Publish}(P, (a, d), \mathcal{G})$: In response, $\texttt{Publish}(pmk, P, (a, d), \mathcal{G})$ is executed using pmk of U_P. $\mathcal{A}$ is then given the modified profile P and all updated keys of corrupted users $U \in \mathcal{C}$.

$\mathsf{Retrieve}(P, a, U)$: In response, $\texttt{Retrieve}(rk_U, P, a)$ is executed using rk_U of U and its output is given back to $\mathcal{A}$.

$\mathsf{Delete}(P, a)$: In response, $\texttt{Delete}(pmk, P, a)$ is executed using pmk of U_P. $\mathcal{A}$ is then given the modified profile P and all updated keys belonging to corrupted users $U \in \mathcal{C}$.

$\mathsf{ModifyAccess}(P, a, U)$: In response, $\texttt{ModifyAccess}(pmk, P, a, U)$ is executed using pmk of U_P. $\mathcal{A}$ is then given the modified profile P and all updated keys belonging to corrupted users $U \in \mathcal{C}$.

3.3 Security and Privacy Requirements

Subsequently, we define two security requirements for a profile management scheme. Our first requirement, called *confidentiality*, aims to protect attributes d stored in a profile from unauthorized access. Our second requirement, called *unlinkability* aims at protecting the privacy of users in the following sense: a profile management scheme should hide information on whether a user U has been authorized to access some attribute in a profile of another user U_P, and, moreover, it shouldn't leak information whether different attributes within a profile can be accessed by the same user, even if the adversary has access to these attributes as well.

Confidentiality. We model *confidentiality* in Definition 2 using the standard indistinguishability approach, similar to the one used in definitions of secure encryption, i.e. it should be computationally infeasible for an adversary $\mathcal{A}$ to decide which attribute d is referenced by an index a.

Definition 2 (Confidentiality). *Let* PMS *be a profile management scheme from Definition 1 and $\mathcal{A}$ be a PPT adversary interacting with users via queries from Section 3.2 within the following game* $\mathsf{Game}^{\mathsf{conf}}_{\mathcal{A},\mathrm{PMS}}$*:*

1. $\texttt{Init}(\kappa)$ *is executed for all users* $U \in \mathcal{U}$.
2. $\mathcal{A}$ *can execute arbitrary operations and ask queries. At some point it outputs* (a, d_0), $(a, d_1) \in \mathcal{I} \times \{0,1\}^*$, $\mathcal{G}_t \subset \mathcal{U}$, *and* $U_P \in \mathcal{U} \setminus \mathcal{G}_t$ *such that neither* U_P *nor any* $U \in \mathcal{G}_t$ *is corrupted (i.e.,* $(\{U_P\} \cup \mathcal{G}_t) \cap \mathcal{C} = \emptyset$*) and* $|d_0| = |d_1|$ *(i.e.,* d_0 *and* d_1 *have the same length).*
3. *Bit* $b \in_R \{0,1\}$ *is chosen uniformly,* $\texttt{Publish}(pmk, P, (a, d_b), \mathcal{G}_t)$ *with* pmk *of* U_P *is executed, and the modified* P *is given to* $\mathcal{A}$.
4. $\mathcal{A}$ *can execute arbitrary operations and ask queries. At some point it outputs bit* $b' \in \{0,1\}$.
5. $\mathcal{A}$ *wins, denoted by* $\mathsf{Game}^{\mathsf{conf}}_{\mathcal{A},\mathrm{PMS}} = 1$*, if all of the following holds:*
   - $b' = b$,
   - $U_P \notin \mathcal{C}$,
   - $\mathcal{A}$ *did not query* $\mathsf{Retrieve}(P, a, U)$ *with* $U \in \mathcal{G}^*_{a,P}$.
   - $\mathcal{G}^*_{a,P} \cap \mathcal{C} = \emptyset$ *(users that have ever been authorized to access the attribute indexed by* a *in* P *are not corrupted).*

The advantage probability of $\mathcal{A}$ *in winning the game* $\mathsf{Game}^{\mathsf{conf}}_{\mathcal{A},\mathrm{PMS}}$ *is defined as*

$$\mathsf{Adv}^{\mathsf{conf}}_{\mathcal{A},\mathrm{PMS}}(\kappa) := \left| \Pr\left[\mathsf{Game}^{\mathsf{conf}}_{\mathcal{A},\mathrm{PMS}} = 1\right] - \frac{1}{2} \right|$$

We say that PMS *provides* confidentiality *if for any PPT adversary* $\mathcal{A}$ *the advantage* $\mathsf{Adv}^{\mathsf{conf}}_{\mathcal{A},\mathrm{PMS}}(\kappa)$ *is negligible.*

Unlinkability. We model *unlinkability* in Definition 3 using the indistinguishability approach as well, but this time $\mathcal{A}$ must decide which user, U_0 or U_1, has been authorized (via `Publish` or `ModifyAccess`) to access an attribute in a profile P even if $\mathcal{A}$ has access to P. Our definition also models access unlinkability of users across different profiles, e.g. P and P', since $\mathcal{A}$ can corrupt the owner of any other profile P' and thus learn all secrets that U_0 and U_1 might possess in these other profiles.

Definition 3 (Unlinkability). *Let* PMS *be a profile management scheme from Definition 1 and $\mathcal{A}$ be a PPT adversary interacting with users via queries from Section 3.2 within the following game* $\mathsf{Game}^{\mathsf{unlink}}_{\mathcal{A},\mathrm{PMS}}$*:*

1. $\texttt{Init}(\kappa)$ *is executed for all users* $U \in \mathcal{U}$.
2. $\mathcal{A}$ *can execute arbitrary operations and ask queries. At some point it outputs* U_0*,* U_1*,* (a, d)*, and* U_P *(owner of some profile* P*).*

3. *Bit* $b \in_R \{0,1\}$ *is chosen uniformly and:*
   - *If* $(a,*) \notin P$ *then* $\texttt{Publish}(pmk, P, (a,d), \{U_b\})$ *with* pmk *of* U_P *is executed.*
   - *If* $(a,*) \in P$ *then* $\texttt{ModifyAccess}(pmk, P, a, \{U_b\})$ *with* pmk *of* U_P *is executed.*

   $\mathcal{A}$ is given the modified P and the possibly updated retrieval keys rk_U for all $U \in \mathcal{C}$.
4. *$\mathcal{A}$ can execute arbitrary operations and ask queries. At some point it outputs a bit $b' \in \{0,1\}$.*
5. *$\mathcal{A}$ wins, denoted by* $\mathsf{Game}^{\mathsf{unlink}}_{\mathcal{A},\mathrm{PMS}} = 1$*, if all of the following holds:*
   - $b' = b$*,*
   - $\{U_0, U_1, U_P\} \cap \mathcal{C} = \emptyset$*,*
   - *$\mathcal{A}$ neither queried* $\mathsf{Retrieve}(P, a, U_0)$ *nor* $\mathsf{Retrieve}(P, a, U_1)$*.*

The advantage probability of $\mathcal{A}$ in winning the game $\mathsf{Game}^{\mathsf{unlink}}_{\mathcal{A},\mathrm{PMS}}$ *is defined as*

$$\mathsf{Adv}^{\mathsf{unlink}}_{\mathcal{A},\mathrm{PMS}}(\kappa) := \left| \Pr\left[\mathsf{Game}^{\mathsf{unlink}}_{\mathcal{A},\mathrm{PMS}} = 1\right] - \frac{1}{2} \right|$$

We say that PMS *provides* unlinkability *if for any PPT adversary $\mathcal{A}$ the advantage* $\mathsf{Adv}^{\mathsf{unlink}}_{\mathcal{A},\mathrm{PMS}}(\kappa)$ *is negligible.*

4 Private Profiles with Shared Keys

Our first construction, called PMS-SK, is simple and uses *shared keys* to encrypt profile attributes for a group of authorized users. An independent symmetric key K_a is chosen by the owner of a profile P for each pair (a,d) and distributed to the group $\mathcal{G}$ of users that are authorized to access d. The key is updated on each modification of $\mathcal{G}$. We use a *symmetric encryption scheme* $SE = (SE.KGen, SE.Enc, SE.Dec)$ for which we assume classical indistinguishability against chosen-plaintext attacks (IND-CPA) and denote by $\mathsf{Adv}^{\mathsf{IND-CPA}}_{\mathcal{A},SE}(\kappa)$ the corresponding advantage of the adversary.

The distribution of K_a may be performed in two ways: K_a can be communicated to the authorized users online (over secure channels) or offline, e.g., by storing K_a securely (possibly using asymmetric encryption) either within the profile or at some centralized server. Our specification of PMS-SK leaves open how distribution of shared keys is done. In particular, the use of one or another technique may be constrained by the application that will use the scheme.

4.1 Specification of PMS-SK

In our constructions we implicitly assume that the uniqueness of indices a in a profile P is implicitly ensured or checked by corresponding algorithms.

$\texttt{Init}(\kappa)$: Output $P \leftarrow \emptyset$ and $pmk \leftarrow \emptyset$.

$\texttt{Publish}(pmk, P, (a,d), \mathcal{G})$: $K_a \leftarrow SE.KGen(1^\kappa)$, add $(a, SE.Enc(K_a, d))$ to P, K_a to rk_U for each $U \in \mathcal{G}$, and K_a to pmk.

$\texttt{Retrieve}(rk_U, P, a)$: Extract K_a from rk_U. If $(a, \bar{d}) \in P$ for some $\bar{d}$ then output $SE.Dec(K_a, \bar{d})$, else $\perp$.

`Delete`(pmk, P, a) : Delete $(a, \bar{d})$ from P. Delete K_a from pmk.
`ModifyAccess`(pmk, P, a, U) : If $U \in \mathcal{G}$ then remove U from $\mathcal{G}$, otherwise add U to $\mathcal{G}$. Execute `Delete`(pmk, P, a) followed by `Publish`$(pmk, P, (a, d), \mathcal{G})$ where d is the attribute indexed by a.

The description of `ModifyAccess` is kept general in the sense that it does not specify how the profile owner U_P reveals an attribute d indexed by a. Our scheme allows for different realizations: d can be stored by U_P locally (not as part of P) or it can be obtained through decryption of $\bar{d}$ using K_a which is part of pmk.

4.2 Complexity Analysis

PMS-SK requires each profile owner U_P to store one key per attribute $(a, \bar{d})$ currently published in P. Additionally, each user has to store one key per attribute she is allowed to access in any profile. Therefore, assuming the worst case where all users in $\mathcal{U}$ have profiles containing $|P|$ attributes that can be accessed by all other users, PMS-SK requires each $U \in \mathcal{U}$ to store $N \cdot |P|$ keys from which $|P|$ keys are stored in its pmk and $(N-1) \cdot |P|$ in the retrieval keys rk_U for all others users' profiles. For each `Publish` or `ModifyAccess` operation the profile owner needs further to perform one symmetric encryption.

4.3 Security and Privacy Analysis

In this section we prove that PMS-SK ensures confidentiality of attributes and provides unlinkability for the authorized users. (See full version for the proof of Theorem 1.)

Theorem 1 (Confidentiality of PMS-SK). *If SE is IND-CPA secure, then* PMS-SK *provides confidentiality from Definition 2, and*

$$\mathsf{Adv}^{\mathsf{conf}}_{\mathcal{A},\text{PMS-SK}}(\kappa) \leq (1+q) \cdot \mathsf{Adv}^{\mathsf{IND-CPA}}_{\mathcal{A}^*,SE}(\kappa)$$

with q being the number of invoked `ModifyAccess` *operations per attribute.*

Theorem 2 (Unlinkability of PMS-SK). PMS-SK *provides perfect unlinkability as defined in Definition 3, i.e.,* $\mathsf{Adv}^{\mathsf{unlink}}_{\mathcal{A},PMS-SK}(\kappa) = 0$.

Proof. The attribute keys K_a are statistically independent of the identities of users in $\mathcal{G}$ who have been authorized to access the attribute indexed by a. Therefore, $\mathcal{A}$ cannot win in $\mathsf{Game}^{\mathsf{unlink}}_{\mathcal{A},\text{PMS}}$ better than by a random guess, i.e., with probability $\frac{1}{2}$. □

Remark 1. The perfect unlinkability property of our PMS-SK construction proven in the above theorem should be enjoyed with caution when it comes to the deployment of the scheme in practice. The reason is that PMS-SK does not specify how shared keys are distributed, leaving this to the application that will use the scheme. One approach to distribute keys in a privacy-preserving manner is given by Barth, Boneh, and Waters [2] and the *CCA recipient privacy* of their scheme, which however comes with storage overhead linear in the number of recipients and may be undesirable when encrypting small-sized attributes in social profiles. In any case it is clear that the distribution process will eventually have impact on the unlinkability property of the scheme, maybe to the point of ruling out its perfectness.

4.4 Further Optimizations

Regardless of the question, whether shared keys K_a are distributed by the application in an online or an offline fashion, there is a way to further optimize and further improve the actual management of these keys. In our specification of PMS-SK these keys are currently chosen fresh for each modification of the authorized group $\mathcal{G}$. However, by using *group key management schemes* that allow efficient update of group keys such as LKH [23,22] or OFT [6,20,17] with all the resulting efficiency differences, the overhead for the distribution can be further reduced.

Another optimization concerns generation of shared keys K_a in case a profile owner U_P does not wish to store corresponding attributes d (outside of the profile). Instead of storing linear (in the number of attributes in P) many shared keys in pmk, the profile owner can derive each K_a using some pseudorandom function $f_s(a, i)$ where s is a seed used for all attributes, a is the unique attribute index, and i is a counter that is updated on each execution of `ModifyAccess` on a to account for possible repetitions of the authorized group $\mathcal{G}$ over the life time of the profile. This optimization allows to trade in the storage costs for pmk for the computation overhead for deriving K_a.

We do not analyze the efficiency effects of the proposed optimizations in detail here, as the construction based on broadcast encryption presented in the next section has only a constant overhead of retrieval keys.

5 Private Profiles with Broadcast Encryption

Our second generic construction of a profile management scheme, called PMS-BE, is based on an adaptively secure (identity-based) broadcast encryption scheme, e.g. [9], whose syntax and requirements we recall in the following.

Definition 4 (Broadcast Encryption Scheme [9]). *A* broadcast encryption scheme $BE = (BE.Setup, BE.KGen, BE.Enc, BE.Dec)$ *consists of the following algorithms:*

$BE.Setup(\kappa, n, \ell)$: *On input the security parameter κ, the number of receivers n, and the maximal size $\ell \leq n$ of the recipient group, this probabilistic algorithm outputs a public/secret key pair $\langle PK, SK \rangle$.*

$BE.KGen(i, SK)$: *On input an index $i \in \{1, \ldots, n\}$ and the secret key SK, this probabilistic algorithm outputs a private (user) key sk_i.*

$BE.Enc(S, PK)$: *On input a subset $S \subseteq \{1, \ldots, n\}$ with $|S| \leq \ell$ and a public key PK, this probabilistic algorithm outputs a pair $\langle Hdr, K \rangle$ where Hdr is called the header and $K \in \mathcal{K}$ is a message encryption key.*

$BE.Dec(S, i, sk_i, Hdr, PK)$: *On input a subset $S \subseteq \{1, \ldots, n\}$ with $|S| \leq \ell$, an index $i \in \{1, \ldots, n\}$, a private key sk_i, a header Hdr, and the public key PK, this deterministic algorithm outputs the message encryption key $K \in \mathcal{K}$.*

Correctness of BE requires that for all $S \subseteq \{1, \ldots, n\}$ and all $i \in S$, if $\langle PK, SK \rangle \leftarrow_R BE.Setup(\kappa, n, \ell)$, $sk_i \leftarrow_R BE.KGen(i, SK)$, and $\langle Hdr, K \rangle \leftarrow_R BE.Enc(S, PK)$, then $BE.Dec(S, i, sk_i, Hdr, PK) = K$.

The adaptive security of BE against chosen plaintext attacks as defined in [9] can be extended to chosen-ciphertext attacks as follows.

Definition 5 (Adaptive CCA-Security of BE). *Let BE be a broadcast encryption scheme from Definition 4 and $\mathcal{A}$ be a PPT adversary in the following game, denoted $\mathsf{Game}^{\mathsf{ad-CCA}}_{\mathcal{A},BE,n,\ell}(\kappa)$:*

1. *$\langle PK, SK\rangle \leftarrow_R BE.Setup(\kappa, n, \ell)$. $\mathcal{A}$ is given PK (together with n and ℓ).*
2. *$\mathcal{A}$ adaptively issues private key queries $BE.KGen(i)$ for $i \in \{1, \ldots, n\}$ and obtains corresponding sk_i. In addition, $\mathcal{A}$ is allowed to query $BE.Dec(S, i, Hdr, PK)$ to obtain message encryption keys K.*
3. *$\mathcal{A}$ outputs a challenge set of indices S^*, such that no $BE.KGen(i)$ with $i \in S^*$ was asked. Let $\langle Hdr^*, K_0\rangle \leftarrow_R BE.Enc(S^*, PK)$ and $K_1 \in_R \mathcal{K}$. Bit $b \in_R \{0, 1\}$ is chosen uniformly and $\mathcal{A}$ is given (Hdr^*, K^*) with $K^* = K_b$.*
4. *$\mathcal{A}$ is allowed to query $BE.Dec(S, i, Hdr, PK)$, except on inputs of the form $\langle S^*, i, Hdr^*, PK\rangle$, $i \in S^*$.*
5. *$\mathcal{A}$ outputs bit $b' \in \{0, 1\}$ and wins the game, denoted $\mathsf{Game}^{\mathsf{ad-CCA}}_{\mathcal{A},BE,n,\ell}(\kappa) = 1$, if $b' = b$.*

We define $\mathcal{A}$'s advantage against the adaptive CCA-security of BE as

$$\mathsf{Adv}^{\mathsf{ad-CCA}}_{\mathcal{A},BE,n,\ell}(\kappa) = \left|\Pr\left[\mathsf{Game}^{\mathsf{ad-CCA}}_{\mathcal{A},BE,n,\ell}(\kappa) = 1\right] - \frac{1}{2}\right|$$

We say that BE is adaptively CCA-secure *if for all PPT adversaries $\mathcal{A}$ the advantage $\mathsf{Adv}^{\mathsf{ad-CCA}}_{\mathcal{A},BE,n,\ell}(\kappa)$ is negligible.*

5.1 Specification of PMS-BE

The main idea behind PMS-BE is that each profile owner U_P manages independently its own instance of the BE scheme in the following way: U_P assigns fresh indices i, which we call *pseudonyms*, to the users from $\mathcal{U}$ (upon their first admission to P) and gives them corresponding private (user) keys sk_i. In order to publish an attribute d for some authorized group $\mathcal{G}$, the owner encrypts d using the BE scheme and the set of indices assigned to the users in $\mathcal{G}$. This process allows for very efficient modification of the authorized group $\mathcal{G}$: In order to admit or remove a member U with regard to d the profile owner simply adjusts $\mathcal{G}$ and re-encrypts d. In particular, there is no need to distribute new decryption keys. However, this flexibility comes at the price of a somewhat weaker privacy, since BE schemes include indices i into ciphertext headers, which in turn allows for linkability of an authorized user U across multiple attributes within P. Yet, the use of pseudonyms still allows us to show that PMS-BE satisfies the weaker requirement of *anonymity*, which we introduce and formally relate to unlinkability in Section 5.4.

`Init`(κ) : Execute $\langle PK, SK\rangle \leftarrow BE.Setup(\kappa, n, \ell)$ with $n = \ell = N$ [3]. Output $P \leftarrow \emptyset$ and $pmk \leftarrow \{PK, SK\}$. Additionally, PK is made public.

[3] We use this upper bound for simplicity here. One may cut down both on n and ℓ to improve the efficiency of BE.

`Publish`$(pmk, P, (a, d), \mathcal{G})$: For every $U \in \mathcal{G}$ without pseudonym for P pick an unused pseudonym i at random from $[1, n]$, extract $sk_i \leftarrow BE.KGen(i, SK)$, and define new $rk_U \leftarrow \langle i, sk_i \rangle$. For every $U \in \mathcal{G}$ add the corresponding pseudonyms to the set S. Compute $\langle Hdr, K_a \rangle \leftarrow BE.Enc(S, PK)$, $\hat{d} \leftarrow SE.Enc(K_a, d)$, and $\bar{d} \leftarrow \langle Hdr, S, \hat{d} \rangle$. Add $(a, \bar{d})$ to P and K_a to pmk. Output P, all new rk_U, and pmk.

`Retrieve`(rk_U, P, a) : Extract $(a, \bar{d})$ from P. Parse $\bar{d}$ as $\langle Hdr, S, \hat{d} \rangle$. Extract $\langle i, sk_i \rangle$ from rk_U. Set $K_a \leftarrow BE.Dec(S, i, sk_i, Hdr, PK)$ and output $SE.Dec(K_a, \hat{d})$.

`Delete`(pmk, P, a) : Delete $(a, \bar{d})$ from P. Delete K_a from pmk.

`ModifyAccess`(pmk, P, a, U) : If $U \in \mathcal{G}$ remove U from $\mathcal{G}$; otherwise add U to $\mathcal{G}$. Execute `Delete`(pmk, P, a) followed by `Publish`$(pmk, P, (a, d), \mathcal{G})$, where d is the attribute indexed by a.

5.2 Complexity Analysis

PMS-BE requires each profile owner U_P to store one key per index-attribute pair $(a, \bar{d})$ currently published in P as well as the key pair $\langle PK, SK \rangle$. For each profile containing at least one attribute a user is allowed to access, this user has to store its secret key $\langle i, sk_i \rangle$ contained in rk_U. Assuming the worst case where all users in $\mathcal{U}$ have profiles containing $|P|$ attributes that can be accessed by all other users, PMS-BE requires each $U \in \mathcal{U}$ to store $|P| + N + 1$ keys from which $|P| + 2$ keys are stored in pmk and $N - 1$ secret keys $\langle i, sk_i \rangle$ are stored in the retrieval keys rk_U of all others users' profiles. For each `Publish` or `ModifyAccess` operation the profile owner needs further to perform one broadcast encryption $BE.Enc$ and one symmetric encryption.

The storage overhead may be reduced by omitting the storage of attribute keys K_a in pmk as the profile owner is able to reconstruct K_a by executing $sk_i \leftarrow BE.KGen(i, SK)$ for any index i in the set of authorized indices S for a. With the authorized user's secret key sk_i, the profile owner is able to execute $BE.Dec$, receiving K_a. That way, the total number of stored keys is reduced by $|P|$ to $N + 1$, traded in for a higher computation overhead when executing `ModifyAccess`.

Obviously, the main advantage of the PMS-BE construction over the PMS-SK approach is the number of keys that have to be stored in rk_U, which is only one in PMS-BE whereas PMS-SK imposes key overhead linear in the number of attributes. However, this efficiency benefit comes at the cost of a weaker privacy, as we discuss below.

5.3 Confidentiality of PMS-BE

We first analyze the confidentiality property of the PMS-BE scheme. (See full version for the proof of Theorem 3.)

Theorem 3 (Confidentiality of PMS-BE). *If SE is IND-CPA secure and BE is adaptively CCA-secure,* PMS-BE *provides confidentiality from Definition 2, and*

$$\mathsf{Adv}^{\mathsf{conf}}_{\mathcal{A},\text{PMS-BE}}(\kappa) \leq (1 + q) \cdot \left(\mathsf{Adv}^{\mathsf{IND-CPA}}_{\mathcal{B}_1, SE}(\kappa) + N \cdot \mathsf{Adv}^{\mathsf{ad-CCA}}_{\mathcal{B}_2, BE, n, \ell}(\kappa) \right)$$

with q being a number of invoked `ModifyAccess` *operations per attribute.*

5.4 Privacy of PMS-BE

Our PMS-BE construction does *not* provide unlinkability as defined in Definition 3 since the indices of users are linkable across different published attributes. The attack is simple: After initialization of PMS-BE, unlinkability adversary $\mathcal{A}$ outputs two arbitrary users U_0, U_1, some pair (a, d) and a profile owner U_P. Then, $\mathcal{A}$ executes $\mathsf{Publish}(P, (a', d'), \{U_0\})$ for an arbitrary pair (a', d') and extracts the two pairs $(a, \bar{d})$ and $(a', \bar{d}')$ from P such that $\bar{d} = \langle Hdr, S, \hat{d} \rangle$ and $\bar{d}' = \langle Hdr', S', \hat{d}' \rangle$). If $S = S'$, $\mathcal{A}$ outputs 0, otherwise 1.

Since PMS-BE has simpler management and distribution of retrieval keys it would be nice to see whether it can satisfy some weaker, yet still meaningful privacy property. It turns out that PMS-BE is still able to provide *anonymity* of users that are members of different authorized groups $\mathcal{G}$ within the same profile, even in the presence of an adversary in these groups.

We formalize anonymity by modifying the unlinkability game based on the following intuition: An anonymity adversary shall not be able to decide the identity of some user $U_b \in \{U_0, U_1\}$ in the setting where the adversary is restricted to publish attributes or modify access to them either by simultaneously including both U_0 and U_1 into the authorized group or none of them. This definition rules out linkability of users based on their pseudonyms, while keeping all other privacy properties of the unlinkability definition.

Finally, we can prove that PMS-BE provides *perfect anonymity* using similar arguments as we used for the perfect unlinkability of the PMS-SK scheme. Nevertheless, we observe that our discussion in Remark 1 regarding the potential loss of perfectness for the unlinkability of PMS-SK when deployed in the concrete application applies to PMS-BE as well, due to the distribution of private user keys sk_i.

(See full version for the formal definition of anonymity and for the proofs that unlinkability is strictly stronger than anonymity and PMS-BE is perfectly anonymous.)

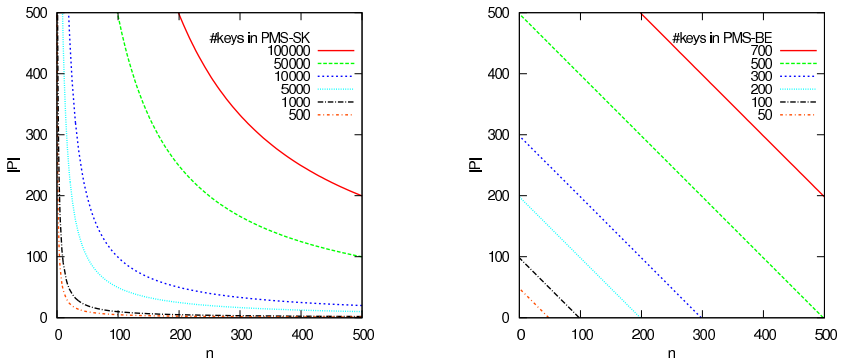
6 Analysis and Discussion for Real-World Social Communities

We analyze the impact imposed by PMS-SK and PMS-BE schemes on the most representative online social community *Facebook* as well as the two well-known services *Twitter* and *Flickr*, and focus on the main complexity difference between both approaches, namely on the average overhead for the storage of private keys.

Being a very general platform for social networking, Facebook users share data with a high amount of contacts. Facebook's own statistics[4] indicates an average of 130 contacts per user, while Golder et al. [11] found a mean of about 180. According to Facebook's statistics, about 500 million active users share more than 30 billion pieces of content (e.g., web links, blog posts, photo albums, etc.) each month. Assuming a rather short lifetime of only three months per item, each user stores on average about 180 pieces of content, i.e., attributes in our profile management scheme.

Assuming an average of 150 contacts per user and 180 attributes per profile we obtain 332 keys that have to be stored by each user when using PMS-BE in contrast to

[4] `http://www.facebook.com/press/info.php?statistics`, Oct 2010



(a) PMS-SK (b) PMS-BE

Fig. 1. Plots of the number of keys each user has to store in PMS-SK resp. PMS-BE, depending on the average number of contacts n and the average number of attributes per profile $|P|$.

over 27000 keys that would be required by PMS-SK. Considering a key length of 192 bits for the private (decryption) key as a basis, this results in a storage overhead of about 8 KB for PMS-BE, compared to about 650 KB for PMS-SK.

Regarding the microblogging service Twitter, where users have on average approximately 50 contacts ("followers") and publish about 60 attributes ("tweets") per month[5], the number of stored keys per user is 232 in PMS-BE and over 9000 in PMS-SK, resulting in about 6 KB respectively 220 KB storage overhead (assuming again a lifetime of three months). Flickr, an online community for image and video hosting, has a very low average of only 12 contacts ("friends") per user according to a study of Mislove et al. [18] in 2007. Assuming the limit of 200 images for a "free account"[6] as average number of attributes per profile, the number of keys each Flickr user has to store would be 214 in the PMS-BE construction and 2600 in PMS-SK, which yields a storage overhead of about 5 KB respectively 62 KB.

We observe that in these average settings the absolute difference of both approaches in storage overhead is not very high. Although relatively differing by a factor of roughly 100, the absolute storage overhead for the assumed average parameters remains below 1 MB in all three networks. We observe that these costs are practical not only for desktop computers but also for modern smart phones. Hence, in practice the two constructions PMS-SK and PMS-BE would allow for a trade-off between minimization of the storage overhead and maximization of privacy (as PMS-BE only provides anonymity, but not unlinkability).

On the other hand, when applied to very large profiles (with more contacts and attributes than assumed above) the difference in storage overhead increases rapidly as illustrated in Figure 1. For example, a profile with 300 contacts[7] and 2000 attributes

[5] `http://www.website-monitoring.com/blog/2010/05/04/twitter-facts-and-figures-history-statistics/`, Oct 2010

[6] `http://www.flickr.com/help/limits/`, Oct 2010

[7] More than 10% of the Facebook users have more than 300 contacts [10].

leads to the overhead of about 15 MB in PMS-SK compared to only 55 KB in PMS-BE. Therefore, using PMS-BE is advisable in this case.

7 Conclusion and Future Work

Privacy preserving publication of personal data in user profiles is a valuable building block that can be used to boot-strap various collaborative and social data sharing applications. So far, security and privacy of user profiles have been addressed in a rather informal way, resulting in several implementations with unclear requirements.

In this work, we gave a rigorous security model for private user profiles, capturing the confidentiality of profile data and privacy of users that are allowed to retrieve this data. Our model allows for the construction of private profile management schemes, independently of the social application that will use them. It aims at local schemes, which can be used both in centralized and distributed environments.

Furthermore, we gave two concrete constructions of profile management schemes, using symmetric and broadcast encryption techniques. Our analysis showed that these constructions differ in the privacy guarantees they provide. While one offers strong privacy with the notion of unlinkability, the other represents a trade-off for better efficiency and key management, yet, consequently offers only anonymity of the users.

A couple of interesting open questions regarding the real-world use of profile management schemes still remains:

- Can unlinkability of users being authorized to access different attributes within a

7. B. Carminati, E. Ferrari, and A. Perego. Enforcing access control in web-based social networks. *ACM Transactions on Information and System Security*, 13(1), 2009.
8. L. A. Cutillo, R. Molva, and T. Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *"IEEE Communications Magazine", Vol 47, Issue 12, Consumer Communications and Networking Series*, 2009.
9. C. Gentry and B. Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In *28th Annual International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT 2009)*, pages 171–188, 2009.
10. M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. Walking in Facebook: A Case Study of Unbiased Sampling of OSNs. In *29th IEEE Conference on Computer Communications (INFOCOM 2010)*, pages 2498–2506, 2010.
11. S. A. Golder, D. M. Wilkinson, and B. A. Huberman. Rhythms of social interaction: Messaging within a massive online network. In *Communities and Technologies 2007*, pages 41–66. 2007.
12. K. Graffi, P. Mukherjee, B. Menges, D. Hartung, A. Kovacevic, and R. Steinmetz. Practical security in p2p-based social networks. In *34th Annual IEEE Conference on Local Computer Networks (LCN 2009)*, pages 269–272, 2009.
13. K. Graffi, S. Podrajanski, P. Mukherjee, A. Kovacevic, and R. Steinmetz. A distributed platform for multimedia communities. In *10th IEEE International Symposium on Multimedia (ISM 2008)*, pages 208–213, 2008.
14. R. Gross and A. Acquisti. Information revelation and privacy in online social networks. In *2005 ACM Workshop on Privacy in the Electronic Society (WPES 2005)*, pages 71–80, 2005.
15. S. Jahid, P. Mittal, and N. Borisov. EASiER: Encryption-based access control in social networks with efficient revocation. In *6th ACM Symposium on Information, Computer and Communications Security (ASIACCS 2011)*. To appear.
16. M. M. Lucas and N. Borisov. flyByNight: mitigating the privacy risks of social networking. In *5th Symposium on Usable Privacy and Security (SOUPS 2009)*, 2009.
17. D. A. McGrew and A. T. Sherman. Key establishment in large dynamic groups using one-way function trees. Technical Report No. 0755, TIS Labs at Network Associates, Inc., Glenwood, MD, May 1998.
18. A. Mislove, M. Marcon, P. K. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *7th ACM SIGCOMM Conference on Internet Measurement 2007*, pages 29–42, 2007.
19. PrimeLife. Scramble! `http://www.primelife.eu/results/opensource/65-scramble`, September 2010.
20. A. T. Sherman and D. A. McGrew. Key establishment in large dynamic groups using one-way function trees. *IEEE Transactions on Software Engineering*, 29(5):444–458, 2003.
21. A. Tootoonchian, S. Saroiu, Y. Ganjali, and A. Wolman. Lockr: better privacy for social networks. In *2009 ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT 2009)*, pages 169–180, 2009.
22. D. Wallner, E. Harder, and R. Agee. Key Management for Multicast: Issues and Architectures. RFC 2627 (Informational), 1999.
23. C. K. Wong, M. G. Gouda, and S. S. Lam. Secure group communications using key graphs. In *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM 1998)*, pages 68–79, 1998.
24. E. Zheleva and L. Getoor. To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In *18th International Conference on World Wide Web (WWW 2009)*, pages 531–540, 2009.
25. Y. Zhu, Z. Hu, H. Wang, H. Hu, and G.-J. Ahn. A collaborative framework for privacy protection in online social networks. Cryptology ePrint Archive, Report 2010/491, 2010. `http://eprint.iacr.org/`.