Protecting Public OSN Posts from Unintended Access

Frederik Armknecht and Manuel Hauptmann
Universitat Mannheim
{armknecht—mhauptma}@uni-mannheim.de

Stefanie Roos and Thorsten Strufe
TU Darmstadt & CASED
{firstname.lastname}@cased.de

ABSTRACT

The design of secure and usable access schemes to personal data represent a major challenge of online social networks (OSNs). State of the art requires prior interaction to grant access. Sharing with users who are not subscribed or previously have not been accepted as contacts in any case is only possible via public posts, which can easily be abused by automatic harvesting for user profiling, targeted spear-phishing, or spamming. Moreover, users are restricted to the access rules defined by the provider, which may be overly restrictive, cumbersome to define, or insufficiently fine-grained.

We suggest a complementary approach that can be easily deployed in addition to existing access control schemes, does not require any interaction, and includes even public, unsubscribed users. It exploits the fact that different social circles of a user share different experiences and hence encrypts arbitrary posts. Hence arbitrary posts are encrypted, such that only users with sufficient knowledge about the owner can decrypt.

Assembling only well-established cryptographic primitives, we prove that the security of our scheme is determined by the entropy of the required knowledge. We consequently analyze the efficiency of an informed dictionary attack and assess the entropy to be on par with common passwords. A fully functional implementation is used for performance evaluations, and available for download on the Web.

1. INTRODUCTION

Profiles and posts in Online Social Networks (OSN) reveal a wealth of personal information about their owners [1]. This is a common, and very immediate threat as users still casually lose control over personal data like addresses and even the state of their health. For instance, compromising photos or personal opinions have been abused for commercial [1] and criminal use [2]. Furthermore, automatic harvesting of profiles and posts by Web servers in data centers yields large scale automatic access: directly at the database by the provider and institutional parties, through API calls by their affiliates, or through simple Web crawlers of arbitrary, external parties. Such mass retrieval amplifies the threats to mass mailing-, phishing-, and scamming campaigns at very large scales[3].

The most common approach for ensuring explicit audience selection is the use of access control mechanisms by configuring provider-defined access rules. However, this automatically denies access to non-members and user who have not interacted within the OSN previously. In consequence, granting access to individuals from the greater social community other than those that explicitly are declared as friends within the OSN, or the intransparent and uncontrolled set of "friends-of-friends", is solely possible by broadcasting a post without any access restriction to the broad public. But such public posts impose the enormous risk of access by entirely unknown and unintended parties, predominantly the afore-mentioned crawlers. Hence, sharing exclusively with the extended social community of a user is not sufficiently supported today.

1.1 Contribution

In this paper, we close this gap by presenting *Partial Knowledge-based Access Control (PKA)*, a non-interactive access control scheme that can be implemented in addition to existing mechanisms. Its main purpose is to protect public posts from unintended access by unknown parties. No out-of-band key exchange is necessary, allowing even friends that are not subscribed to the OSN to access the posts.

The main idea is to encrypt the post using attribute values that are easy to guess for members of a user's social community, yet difficult for strangers and during automated access. As in general even a member of this community may not have knowledge of all values, such a requirement would be too strict in most cases. Instead our scheme provides a higher level of flexibility: For each post, the publisher can *individually* choose n

[1] `http://www.iwf.org.uk/about-iwf/news/post/334-young-people-are-warned-they-may-lose-control-over-their-images-and-videos-once-they-are-uploaded-online`

[2] `http://www.telegraph.co.uk/technology/news/8789538/Most-burglars-using-Facebook-and-Twitter-to-target-victims-survey-suggests.html`

[3] `http://www.theregister.co.uk/2012/10/31/dodgy_brokering_facebook_data/`

attributes as well as the number of attributes a user has to know to access the posts.

We describe a concrete instantiation based on established cryptographic mechanisms. We prove that if these mechansims are secure, the security of *PKA* is completely determined by the entropy of the chosen attributes.

Based on data, we acquired from well-known OSNs and the German statistical office, we find that, even under an advanced and informed dictionary attack, on the order of hundreds of millions of potential attribute combinations have to be tested on average to access profiles that are protected with *PKA*. This is about the same order of effort as it is needed to break passwords [2, 3].

Hence, our scheme protects against common attackers such as scammers and spammers, as well as against casual data loss, and leaks by the provider and governmental agencies. Observe that, depending on the particular use case, the owner may choose attributes of higher entropy, which are harder or virtually impossible to guess for anybody. In fact, the traditional approach of encrypting a post using a shared secret key can be seen as a special case of our approach.

We provide a fully functional Firefox extension for Facebook posts as a prototype, for public download. Locally storing the attribute-value pairs that are used as secrets once they are provided, the profiles containing protected posts can be accessed repeatedly without the need for repeated entry. The overhead of the cryptographic scheme is barely perceived by a legitimate user, yet delays unauthorized access by several days, rendering automatic harvesting by both external parties and social network providers economically unviable.

Section 2 covers the state of the art. We describe the general framework of our approach and give a formal security definition in Section 3. Afterwards, we explain a concrete realization in Section 4. Its security and analysis are analyzed in Sections 5 and 6, respectively, and Section 7 concludes the paper.

2. RELATED WORK

OSN privacy is constantly raising broad interest, and recent years have seen various contributions aiming at preserving it. The approaches differ by their assumed trust model and the adversaries they aim to defend against.

Decentralized social networks, like for instance *Vis-a-Vis* [4], *diaspora* [4], and *Safebook* [5] have been proposed to circumvent the existence of a centralized instance that has a global overview of all subscribers, as well as ubiquitous control. These systems require migration from current services, additionally, their lack of acceptance, partially caused by their lack of competitive functionality and performance, have prevented broad application, so far.

Focusing on confidentiality of content rather than anonymity or unobservability, and thus accepting that acts of communication as well as their participators can be observed by the social network provider, a second class of approaches encrypts content and leverages the conventional service for its exchange. *NOYB* [6] as one of the earliest approaches introduces dictionaries to replace the real with seemingly random attributes. The correct mapping is shared with authorized contacts, who hence are capable of retrieving the plaintext. This mapping can be retrieved by the social networking provider, too, though, when acting on behalf of a user's friend. *Persona* [7] and *EASiER* [8] employ attribute-based encryption and offer fine grained, personalized access rules to different parts of the profile. The Firefox extension Scramble! [9] offers item-specific access rules over multiple social networking sites, but again requires the user to explicitly define friendship relations in several social networks.

Facebook indeed has started to prevent sharing of encrypted content by transcoding image files and restricting free-text attribute fields. Some recent approaches hence suggest the exchange of encrypted content via third party storage, using the OSN only for exchange of references to the encrypted content [10, 11, 12, 13]. Alternatively, the transmission of encrypted content can be hidden [14]. All these approaches require interaction for explicit authorization on a per-user base and key exchange.

Approaches that aim at preventing unnoticed access by unintended parties again differ in the adversary they assume. PoX [15] is a plugin that illustrates which information is provided to third-party applications. It additionally offers the possibility to restrict access of these parties to selected profile entries. Our goal differs from theirs: they do not actually want to hide content from general, unintended audiences, but rather enable the user to consciously decide which data is accessed by application providers. *CAPTCHAs* indeed serve the purpose of preventing automated mass retrieval, but they allow access to arbitrary human individuals that can solve the challenge, thus not only to trusted parties. In a similar fashion, text in OSNs can be distorted to protect against automatic processing [16]. Using personal knowledge of and about a user to complement keys and passphrases is leveraged by *social authentication*[5] to prevent illegitimate login to accounts. Both CAPTCHAs and social authentication are aimed at adversaries external to the service, though, and thus fail to prevent unintended access by the provider and its affiliates.

In summary, there are various protection schemes

[4]http://joindiaspora.com

[5]http://www.facebook.com/blog/blog.php?post=486790652130

that can be used to enhance privacy in OSNs. In general, they require explicit audience selection, which is desired in case of highly sensitive data. However, these access rules exclude non-members and may either be overly restrictive or entail over-sharing due to the heterogeneous contacts summarized under the common term friends or friends-of-friends.

3. FRAMEWORK

In this section, we first give a short, informal overview of our requirements for a knowledge-based access scheme for (public) posts. Afterwards, the required functionalities are formalized, as are our security goals.

3.1 Intended Use

The main goal of our scheme is to protect public post from automatic harvesting. Furthermore, it can also be used to obfuscate information from the social network provider, its affiliates, and curious strangers. However, these might have additional knowledge about a user, so that our security analysis in Section 5.2 does not directly apply. The posts of a user should be readable by any person sharing enough experience with the respective user, regardless of prior interaction within the social network. Because even non-members can access the posts, but not arbitrary strangers, the peer pressure to join a social network for retrieving important information is reduced. Furthermore, it is not necessarily to enlist all people that a user wants to share information with as friends, or sort them into categories, potentially risking confrontation in case of a perceived misclassification. Information can simply be posted as public according to the provided access rules, and be encrypted with our scheme to prevent strangers, especially crawlers for user profiling and spamming, from accessing the data.

Because our scheme is intended to overcome the problems of cumbersome explicit audience selection, transparency and revocation are not considered. If desired, they can be provided by additional access rules. For example, a post can be published within the friends or members group, so that it simply restricts the access to those contacts within the OSN, who indeed know the publisher well.

3.2 Functionality

On a high level, *PKA* allows to transform a post p into a protected post c based on a set of attributes $(a_1, \ldots, a_n)$, where each attribute $a_i = (d_i, v_i)$ is composed of an *attribute description* d_i and an *attribute value* v_i. For example, an attribute description could be 'Name' and an attribute value 'John'. In particular two different attributes $a \neq a'$ may share the same attribute description, e.g., 'Name', but have different values 'John' and 'Jane'. Vice versa, two distinct attributes can have different descriptions, but the same value. A protected post c is published together with the used attribute descriptions $d_1, \ldots, d_n$, and the parameters n and t. Only users who know at least t out of these n attribute values are able to recover the original post p from the protected post c.

For the sake of simplicity, we assume in the following that the values t and n with $t \leq n$ are fixed and publicly known. Let $\mathcal{P}$ be the set of posts, $\mathcal{W}$ be the set of attributes a, $\mathcal{V}$ the set of attribute values v, and $\mathcal{C}$ be the set of all protected posts c.

The main components of *PKA* are the two mechanisms *PROTECT* and *ACCESS*, which allow users to protect their posts and access the posts of another known user, respectively. They are formally defined in Algorithm 1. *PROTECT* takes a post $p \in \mathcal{P}$ and n values for generating the protected post c. The operation *ACCESS* maps t values and the protected post c to a post $\tilde{p}$. In case that at least t values correctly correspond to the values used for protecting post p, $\tilde{p} = p$ is correctly retrieved, and a random post $\tilde{p}$ is returned otherwise.

```
$\textbf{\textit{PROTECT}}$: $\mathcal{V}^n \times \mathcal{P} \to \mathcal{C}$, $((v_1, \ldots, v_n), p) \mapsto c$
$\textbf{\textit{ACCESS}}$: $\mathcal{V}^t \times \{1, \ldots, n\}^t \times \mathcal{C} \to \mathcal{P}$,
        $((\tilde{v}_{i_1}, \ldots, \tilde{v}_{i_t}), (i_1, \ldots, i_t), c) \mapsto \tilde{p}$
        where $c = PROTECT((v_1, \ldots, v_n), p)$
        and $i_1, \ldots, i_t$ distinct
        $\tilde{p} = \begin{cases} p, & \text{if } \tilde{v}_{i_j} = v_{i_j} \text{ for } j = 1 \ldots t \\ \text{a random } r \in \mathcal{P}, & \text{otherwise} \end{cases}$
```

Algorithm 1: Basic Functionalities

3.3 Security Goals

Next, we formalize the notion of security for *PKA*. As confidentiality of the post content is the primary security goal, we adopt an established security notion for symmetric encryption schemes: *indistinguishability under chosen-plaintext attack* (cf. [17]). This is modelled by the following game involving an attacker $\mathcal{A}$ and an hypothetical entity named oracle $\mathcal{O}$:

Step 1: $\mathcal{O}$ samples an n-tuple $(a_1, \ldots, a_n) \in \mathcal{W}$ with $a_i = (d_i, v_i)$ according to some distribution $\mathbb{D}$ and hands the attribute descriptions $(d_1, \ldots, d_n)$ to $\mathcal{A}$.

Step 2: $\mathcal{A}$ generates two different posts $p_0 \neq p_1$ (of same length) and gives these to $\mathcal{O}$.

Step 3: $\mathcal{O}$ flips a random coin $b \in \{0, 1\}$, returns $c = PROTECT((a_i, v_i)_{i=1}^n, p_b)$ to $\mathcal{A}$.

Step 4: $\mathcal{A}$ outputs a bit $b^* \in \{0, 1\}$.

The attacker $\mathcal{A}$ wins the game if $b = b^*$. W.l.o.g., we can assume that $\Pr[b = b^*] \geq 1/2$.[6] Observe that a sim-

[6]If this is not the case we consider a modified adversary $\overline{\mathcal{A}}$ who simply invokes $\mathcal{A}$ and returns the complement $\overline{b^*}$ of the output of $\mathcal{A}$.

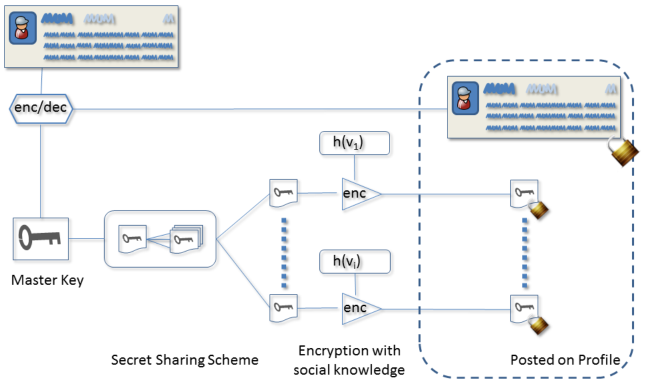


Figure 1: Overview of *PKA* as described in Section 4

ple attack is to pick a random bit for b^*, independent of c, yielding a winning chance of 1/2. Hence the *advantage* $\mathrm{Adv}_{\mathcal{A}}$ of $\mathcal{A}$ is defined by how much the success probability deviates from 1/2:

$$\mathrm{Adv}_{\mathcal{A}} := \Pr[b = b^*] - 1/2. \qquad (1)$$

Here the probability is taken over all random coins of $\mathcal{O}$ and $\mathcal{A}$. Security is then characterized by the maximum advantage taken over all adversaries:

Definition 3.1 (Security). *A* PKA-*scheme is* (τ, ε)-secure *if it holds for all adversaries* $\mathcal{A}$ *with a runtime* $\leq \tau$ *that* $\mathrm{Adv}_{\mathcal{A}} \leq \varepsilon$.

Remark 3.2. *Observe that in the security game, we allow an attacker to see the chosen attribute descriptions* before *it has to choose the two posts* $p_0 \neq p_1$. *This is in compliance with the common approach to make an adversary as strong as possible. On the other hand, we disallow to influence the choice of the attributes. This has several reasons. First, the attributes can be seen as the analogon to the encryption key in symmetric encryption schemes. Second, the security definition should also cover cases where the distribution* $\mathbb{D}$ *is not (or only partially known) to* $\mathcal{A}$. *Third, in case that* $\mathbb{D}$ *has a low entropy, this is reflected accordingly in the success probability of* $\mathcal{A}$.

4. A CONCRETE REALIZATION

In this section, we propose a concrete realization of the *PKA*-concept suggested in Section 3. Our scheme combines established cryptographic primitives:

- a block cipher (Enc, Dec) with key space $\mathcal{K} = \{0,1\}^\kappa$, which encrypts messages (i.e. posts) from $\mathcal{P} = \{0,1\}^b$ to ciphertexts in $C = \mathcal{P}$,[7]
- a hash function $H : \{0,1\}^* \to \mathcal{K}$, and
- a a secret sharing scheme SSS, consisting of two functions: *SSS-CREATE* creates shares $s_1, \ldots, s_n \in \mathcal{K}$ out of a secret $S \in \mathcal{K}$, and S is reconstructed from t shares by *SSS-RECONSTRUCT*.

In a nutshell, the block cipher is used to encrypt the post using a randomly sampled key K. Using the secret sharing scheme, the key K is split into n shares such that any t shares are sufficient for reconstructing K. Each share gets encrypted (using the same block cipher) under individual keys, which are derived from the attribute values v_i and the hash function.

More formally, during the preparation phase attributes $a_1, \ldots, a_n$ as well as integers n and t are chosen. We assume them to be publicly known, e.g. by attaching them to the protected post.

When a user executes $PROTECT((v_1, \ldots, v_n), p)$ for protecting a post p, a master key K is uniformly sampled and p is encrypted to $c = Enc(K, p)$. Using the secret sharing scheme, n shares $(s_1, \ldots, s_n)$ subsequently are generated from K, and the shares s_i are encrypted to $c_i = Enc(H(v_i), s_i)$ based on the hashes $H(v_i)$ of the given attribute values v_i. The encrypted post c is published together with the encrypted shares c_i.

When executing the function *ACCESS* to retrieve the content of an encrypted post c, users provide values $\tilde{v}_{i_j}$ for t of the n attributes. Based on these values, the encrypted shares are then retrieved as follows: for each index i_j, the value $H(\tilde{v}_{i_j})$ is computed and afterwards the share $\tilde{s}_{i_j} := Dec(H(\tilde{v}_{i_j}), s_{i_j})$ is decrypted. A potential key $\tilde{K}$ is obtained from the shares $\tilde{s}_{i_j}$ by SSSreconstruct and the encrypted post is decrypted, i.e. $\tilde{p} = Dec(\tilde{K}, c)$. If the values $\tilde{v}_{i_j}$ have all been correct, then the correct shares s_{i_j} have been determined, the value $\tilde{K}$ is equal to K and in particular $\tilde{p} = p$. Otherwise, that is if at least one of the values $\tilde{v}_{i_j}$ have been wrong, the corresponding value $\tilde{s}_{i_j}$ is wrong with high probability (if the hash function is secure). Due to the properties of the secret sharing scheme, a wrong value $\tilde{K}$ is computed and $\tilde{p}$ will be just a random b-bit string. Both algorithms are displayed in Alg. 2. A formal security analysis is given in the next Section.

5. SECURITY ANALYSIS

This section starts with a proof that the security of the scheme relies purely on the entropy of the selected attributes (if the underlying cryptographic primitives are secure). As a consequence, we analyze a dictionary attack based on the entropy of selected social network data.

5.1 Proof of Security

In this section, we provide an upper bound on the advantage of an attacker. Recall that only established

[7]Remark that fixing the length b of the posts is only done to keep the following description simple. If longer posts need to be encrypted, one can easily adapt the scheme to use the block cipher in an appropriate mode of operation.

```
PROTECT
INPUT: $(v_1, ..., v_n) \in \mathcal{V}^n$, $p \in \mathcal{P}$
    Sample $K \in \mathcal{K}$
    Encrypt $c = Enc(K, m)$
    $(s_1, ..., s_n) = SSS\text{-}CREATE(K)$
    Compute $K_i = H(v_i)$ for $i = 1..n$
    Encrypt $c_i = Enc(K_i, s_i)$ for $i = 1..n$
OUTPUT: $(c, c_1, ..., c_n) \in \mathcal{C}$

ACCESS:
INPUT: $(\tilde{v}_{i_1}, \ldots, \tilde{v}_{i_t}) \in \mathcal{V}^t$, $(i_1, \ldots, i_t)$,
        $cp = (c, ..., c_n) = PROTECT((v_1, ..., v_n), p)$
    Compute $\tilde{K}_{i_j} = H(\tilde{v}_{i_j})$ for $j = 1..t$
    Decrypt $\tilde{s}_{i_j} = Dec(\tilde{K}_{i_j}, c_{i_j})$ for $j = 1..t$
    $\tilde{K} = SSS\text{-}RECONSTRUCT(\tilde{s}_{i_1}, ...\tilde{s}_{i_t})$
    $\tilde{p} = Dec(\tilde{K}, c) = \begin{cases} p, \text{ if } \tilde{v}_{i_j} = v_{i_j} \text{ for } j = 1..t \\ \text{a random } r \in \mathcal{P}, \text{ otherwise} \end{cases}$
OUTPUT: $\tilde{p} \in \mathcal{P}$
```

Algorithm 2: *PROTECT* and *ACCESS* of the proposed concrete realization.

cryptographic primitives, i.e. block cipher, hash function, and secret sharing scheme, are deployed. For each of these concrete realizations are known, which are either secure according to the current state of knowledge, e.g. the block cipher standard AES and the hash function standard SHA-3, or can even be mathematically proven to be secure, e.g. Shamir's secret sharing scheme. Hence, to keep the analysis simple, we assume that these schemes are ideally realized. More precisely, we assume in the following that (i) the block cipher is a random permutation, (ii) the hash function is a random oracle, and (iii) the secret sharing scheme is information-theoretically secure. Moreover, we make the attacker somewhat stronger by considering for the time effort only the number q of queries it makes to the ideal cipher and/or the random oracle.[8]

The main theorem is the following:

THEOREM 5.1. *It holds for any adversary $\mathcal{A}$ that makes at most q queries to the ideal cipher and/or the random oracle that*

$$\text{Adv}_{\mathcal{A}} \leq \frac{3q}{|\mathcal{K}|} + Pr[E_v]. \quad (2)$$

Here E_v denotes the event that $\mathcal{A}$ correctly identifies for the given attribute descriptions $(d_1, \ldots, d_n)$ (see Sec. 3) at least t attribute values.

Proof.

$\mathcal{A}$ knows the encrypted post c, the encrypted shares $c_1, ..., c_n$, and the attribute descriptions $(d_1, \ldots, d_n)$. Let E_{mk} denote the event that during the attack, the attacker made a query to the block cipher using the correct key. In the ideal cipher model, c is equally likely to be an encryption of p_0 and p_1 if $\neg E_{mk}$. Hence

$$\begin{aligned} \text{Adv}_{\mathcal{A}} &= \Pr[b = b^*] - 1/2 \\ &= \Pr[b = b^* | E_{mk}] \cdot \Pr[E_{mk}] \\ &+ \Pr[b = b^* | \neg E_{mk}] \cdot \Pr[\neg E_{mk}] - 1/2 \\ &\leq 1 \cdot \Pr[E_{mk}] + 1/2 \cdot 1 - 1/2 = \Pr[E_{mk}]. \end{aligned}$$

Thus, the advantage of $\mathcal{A}$ is equivalent to the probability $\Pr[E_{mk}]$ of deriving the master key K. Let $s_1, \ldots, s_n$ denote the shares, which encode the key K. Let E_s denote the event that the attacker successfully reconstructed at least t from these n shares. Then it holds

$$\begin{aligned} \Pr[E_{mk}] &= \Pr[E_{mk}|E_s] \cdot \Pr[E_s] + \Pr[E_{mk}|\neg E_s] \cdot \Pr[\neg E_s] \\ &\leq Pr[E_s] + q/|\mathcal{K}| \end{aligned}$$

The second term follows from the fact that $\Pr[E_{mk}|\neg E_s] \leq q/|\mathcal{K}|$. The reason is that if the attacker did not recover sufficient shares, it follows from the information-theoretic security of the secret sharing scheme that the attacker cannot derive any information about K, which he did not know before. Thus, the only remaining option in this case is to coincidentally query the block cipher with the correct key. This is successful with probability at most $q/|\mathcal{K}|$.

Next, we consider $Pr[E_s]$. Recall that the attacker only knows the encryption of the shares, being $c_i = Enc(H(v_i), s_i)$ for the selected attribute values v_i. Let $h_i := H(v_i)$ denote their hash values. We denote by E_h the event that the attacker correctly identified at least t hash values h_i. It follows that

$$\begin{aligned} \Pr[E_s] &= \Pr[E_s|E_h] \cdot \Pr[E_h] + \Pr[E_s|\neg E_h] \cdot \Pr[\neg E_h] \\ &\leq 1 \cdot \Pr[E_h] + q/|\mathcal{K}|. \end{aligned}$$

The rationale behind the second term is similar to above: if an attacker does not know all keys that have been used to encrypt the shares, the best he can do is to guess the remaining ones.

Finally, an upper bound for $Pr[E_h]$ is derived. Let E_v denote the event that the attacker identified at least t values v_i correctly. Using the same line of arguments, it follows that

$$\Pr[E_h] \leq \Pr[E_v] + q/|\mathcal{K}|.$$

Assembling all results yields the claim. □

Theorem 5.1 shows that if a reasonable key size, e.g., 128 bits, are chosen, the advantage of an attacker is dominated by the probability of an attacker of correctly guessing sufficiently many attribute values. Consequently, we consider in the remainder of this section a dictionary attack for guessing these values based on real-world data and distributions for the evaluation.

[8]We stress that these assumptions are only made to keep the analysis simple. In fact, the analysis could easily be adapted to the case that the deployed block cipher and hash function are "only" computationally secure.

5.2 Dictionary Attack Evaluation

Having proven that the security depends solely on the probability of determining the chosen values, dictionary attacks that aim to guess those values are the foremost vulnerability for PKA. We consequently analyze the feasibility of a dictionary attack based on real-world data. Note that this analysis is only valid for automated attacks, most commonly executed by crawlers. The scheme is vulnerable to social engineering attacks, which invest time in retrieving information about the person from out-of-band sources. However, such attacks have a high cost, and are not scalable. Recall that an attacker cannot tell from the fact that he chooses incorrect values, which values have been correct, if any, and which incorrect. Consequently, an unsuccessful attack without additional information about the publisher reveals only that at least $n - t$ values are incorrect. The information gain is negligible, due to the vast number of possible attribute-value combinations. The dictionary attack works as follows: an attacker A uses a *basis distribution* $\mathbb{D}_B$ of attribute values and aims to guess attribute values which have been sampled according to a (possibly unknown) *target distribution* $\mathbb{D}_T$. Based on $\mathbb{D}_B$, A then tries all possible sets of t values in descending order of likelihood. For each guess, a corresponding key candidate $\tilde{K}$ is computed and the encrypted post decrypted using $\tilde{K}$. We assume that A can distinguish correct and incorrect decryptions, since it is easy to recognize real language from randomly generated character strings. Hence, A continues testing different sets of values until the encrypted post is correctly decrypted. In the following, we refer by $\mathcal{A}_t^{\mathbb{D}_B}(\mathbb{D}_T)$ to an attacker on a t-of-4 scheme with target distribution $\mathbb{D}_T$, using basis distribution $\mathbb{D}_B$. Two metrics are important for measuring A's efficiency: the fraction of protected posts that can be accessed using these predefined values, and the cost, meaning the number of trials, A has to invest to find the correct values.

Datasets and Distributions.

We use two available datasets to analyze the efficiency of possible attacks: the first was obtained from meinVZ, a German OSN with strong similarities to Facebook [18], the other from the German business-oriented social network XING [19].

Both datasets contain the attributes descriptions 'first name', 'second name', 'home town', 'ZIP' and 'university'. We chose the $266,804$ profiles that contain all considered attributes from the entire set of 702.986 profiles in meinVZ, and analogously $42,475$ profiles of the overall $756,400$ profiles in XING respectively. Since 'hometown' and 'ZIP' are redundant, we use 'hometown' only, especially as the majority of users decided not to enter their ZIP in XING. To increase the attacker's chances, we preprocessed the data sets by by adding variants of attribute values, for example 'Uni' instead of 'university'. We additionally obtain the overall German statistics as published by the German statistical office [9] (henceforth denoted *SB*).

Basic Distribution	1^{st} Name	2^{nd} Name	Uni	Town
SB	1878	3422	568	688
meinVZ	23139	98013	2861	52989
XING	4590	24305	16423	4686

Table 1: Number of values for each dataset

For our analysis, we assume that each profile contains a post protected with these four respective attributes and that these attributes are independent. Note that information on hometown and university may be publicly available and hence unsuitable choices for the attributes in reality, so they are just used for the purpose of our evaluation for the lack of other data. However, names, places, and institutions are frequently associated with a common past history (name of the favorite teacher, the destination of a school trip, ...), so the respective distributions provide a meaningful analysis of at least a common subset of attribute types.

In the following, $\mathbb{D}_T$(meinVZ) and $\mathbb{D}_T$(XING) denote target distributions, and $\mathbb{D}_B$(meinVZ) and $\mathbb{D}_B$(XING) denote the basic distributions obtained from meinVZ and XING by considering each attribute distribution independently. For the basis distribution, we calculate the frequency distributions of the attribute values for meinVZ, XING and SB, respectively. Table 1 displays the number of different attribute values for each data set.

The order of likelihood is evaluated as follows: For a fixed attribute description, a possible attribute value has rank r with regard to the basis distribution $\mathbb{D}_B$ if it is the r-th most likely value according to distribution $\mathbb{D}_B$. Hence, for given attribute descriptions $d = (d_1, \dots, d_t)$, the rank of a possible assignment of attribute values $v = (v_1, \dots, v_t)$ is estimated as the product of the ranks of the individual v_i's, that is $rank(v)$ is estimated by $\prod_{i=1}^{n} rank(v_i)$. Note that this is a lower bound. We compute estimates only, since it is computationally demanding to compute and store a distribution over several billions of values.

Success rate.

For a set of posts, a basis distribution $\mathbb{D}_B$ and a threshold t, the *success rate* is given by the fraction of posts that can be accessed. The fractions of accessible posts for each set of posts and each distribution are given in Table 2. Naturally, an attacker of the form $\mathcal{A}_t^{\mathbb{D}_B(\text{OSN})}(\mathbb{D}_T(\text{OSN}))$ can access all posts as the basis distribution contains all attribute values occurring in the target distribution. Moreover, as expected the suc-

[9] https://www.destatis.de/DE/Startseite.html

Target Dist.	t	Basic Distribution SB	meinVZ	XING
meinVZ	1	0.6390	1.0	0.9714
	2	0.1923	1.0	0.8172
	3	0.0162	1.0	0.4804
	4	0.0	1.0	0.1339
XING	1	0.7968	0.9973	1.0
	2	0.3612	0.9604	1.0
	3	0.0543	0.6880	1.0
	4	0.0	0.0746	1.0

Table 2: Fraction of accessed posts for different distributions

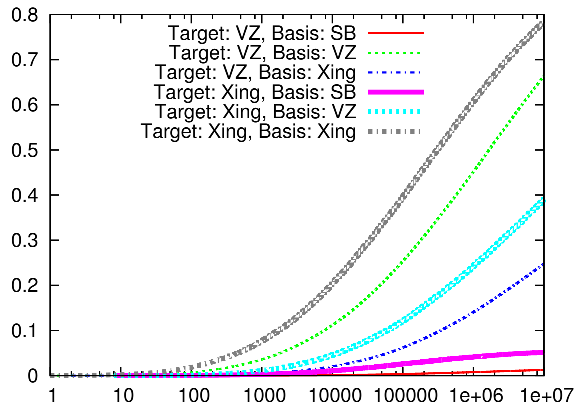


Figure 2: Number of needed trials vs. fraction of accessed posts for a 3-of-4 scheme

cess rate is the lowest if using $\mathbb{D}_B(\text{SB})$ as $\mathbb{D}_B(\text{SB})$ contains less values than the other basis distributions. Even with a threshold of $t = 1$, $\mathcal{A}_1^{\mathbb{D}_B(\text{SB})}(\mathbb{D}_T(\text{meinVZ}))$ only has a success rate of 64 %, whereas $\mathcal{A}_1^{\mathbb{D}_B(\text{SB})}(\mathbb{D}_T(\text{XING}))$ has a success rate of about 80 %. This means that a considerable fraction of the users did not enter a single value that exists in official statistics. The success rate decreases rapidly with increasing t, so that for $t = 4$, no post can be accessed.

Using a basic distribution crawled from a different OSN, more than 80% of the posts can be accessed using a 1-of-4 or 2-of-4 secret sharing scheme. However, when requiring all 4 attributes, we have a success rate of 7.46 % for $\mathcal{A}_4^{\mathbb{D}_B(\text{XING})}(\mathbb{D}_T(\text{meinVZ}))$ and of 13.39 % for $\mathcal{A}_4^{\mathbb{D}_B(\text{meinVZ})}(\mathbb{D}_T(\text{XING}))$, respectively. So, as expected, for both SB and OSN basic distributions, the success rate decreases enormously when increasing t. Though a considerable number of posts can be accessed with a threshold of 1 or 2, less than 15 % of the posts can be accessed with $t \geq 3$.

Mean Number of Trials.

The mean cost of an attack is given as the estimated number of trials, averaged over all posts, with regard to a basic distribution $\mathbb{D}_B$ and a threshold t. Table 3 details the average cost together with the standard deviation. In case of $\mathbb{D}_B(\text{SB})$, the cost for $t = 1 \ldots 3$ lies on the order of at least 10^{11} while the cost for 4-of-4 equals the number of possible combinations (about $2 * 10^{12}$), because no post can be accessed. Nevertheless, this is lower than the cost for $\mathcal{A}_t^{\mathbb{D}_B(\text{meinVZ})}(\mathbb{D}_T(\text{XING}))$ and $\mathcal{A}_t^{\mathbb{D}_B(\text{XING})}(\mathbb{D}_T(\text{meinVZ}))$ for $t = 1 \ldots 4$. The reason for this lies again in the lower number of values considered by the SB basic distribution. In both cases, the cost is at least on the order of 10^{14}, even for $t = 1$. The cost when using XING as a basic distribution and meinVZ as a target distribution is lower than in the opposite case, due to the fact that $\mathbb{D}_B(\text{SB})$ considers less values. In case of an attack of the form $\mathcal{A}_t^{\mathbb{D}_B(\text{OSN})}(\mathbb{D}_T(\text{OSN}))$, the cost is considerably reduced (see Table 3).

Payoff Success vs. Invested Time.

The mean cost is only of limited value, especially since the standard deviation is high. An adversary is more interested in minimizing the effort for maximum success. Hence, A will only try a certain number of values before aborting and trying a different post. This allows us to compare different distributions with regard to their efficiency. Figure 2 displays the effort in terms of needed of trials versus the likelihood of accessing a post using up to 10^7 trials (for higher values the increase is barely noticeable). Indeed, the curves have a strong increase in the beginning, as expected. Using the same basic and target distribution is more efficient than using data from a different OSN. However, the latter still produces a higher success rate for any number of trials than the SB distribution. This is due to the frequent use of inofficial names in profiles, such as storing the nickname as a users first name. Figure 2 indicates that XING posts can be accessed within a lower number of trials, at least for $t = 3$. A plausible reason is that XING is addressed to professionals, meaning users in general use less inofficial names, so having more common attribute values and a smaller set of values.

Summary.

For a threshold $t = 3$, at least 100 million trials are needed in average to access a post. Though this can be realized, it is not within the computational power of a spammer or an advertisement company, providing a good security against this kind of adversary. Moreover, the entropy of our scheme seems to be similar to that of actual selected passwords [2, 3].

Recall that our analysis assumes that values with a low entropy are chosen. As proven in Th. 5.1, the security depends solely on the entropy of the selected attribute values. That is already a 1-of-1 scheme with an arbitrary character string as value is secure, meaning

		Basic Distribution		
Target Dist.	t	SB	meinVZ	XING
meinVZ	1	9.06E11 ± 1.21E12	6.53E01 ± 1.52E02	2.45E14 ± 1.43E15
	2	2.03E12 ± 9.90E11	1.14E05 ± 9.73E05	1.57E15 ± 3.32E15
	3	2.47E12 ± 3.17E11	2.24E09 ± 2.98E10	4.46E15 ± 4.29E15
	4	2.51E12 ± 0.00E00	1.30E14 ± 1.93E15	7.44E15 ± 2.92E15
XING	1	5.10E11 ± 1.01E12	9.15E14 ± 1.77E16	4.98E01 ± 2.01E02
	2	1.60E12 ± 1.21E12	1.36E16 ± 6.70E16	8.27E04 ± 6.75E05
	3	2.37E12 ± 5.69E11	1.07E17 ± 1.59E17	5.84E08 ± 6.13E09
	4	2.51E12 ± 0.00E00	3.18E17 ± 9.03E16	9.13E12 ± 1.06E14

Table 3: Mean number of trials needed to access a post using a dictionary attack for three distinct value distributions (E notation)

it is not possible to determine the protected content. In this fashion, our scheme could be used in the same way as traditional encryption, providing the same guarantees as any well-established cryptographic primitives. At the end, it is up to the concrete scenario to choose a balance between security and probability that a legitimate user can determine the requested attribute values.

6. IMPLEMENTATION

To give proof of our concept, we implemented *PKA*, employing Shamir's secret sharing [20], AES-256, and the SHA-256 hash function, as a Firefox extension in JavaScript. It is available for download [10] and it can be used to publish protected posts in Facebook notes, through a simple graphical user interface. We evaluate the performance of our implementation of *PKA* with regard to the overhead for key generation, encryption and decryption. The evaluation shows that the overhead is negligible for an informed user, but prohibitive for an adversary.

6.1 Extension Description

The extension directly implements the *PROTECT* and *ACCESS* functions. More precisely, the interface provides input methods to share and retrieve protected posts, including the definition of the parameters n and t. It further allows for addition, modification, and deletion of attribute-value pairs, which are stored locally. The interface of the extension is defined using XUL, and all algorithms are implemented in JavaScript, relying on existing, verified libraries. All message exchange leverages the Facebook developer API, and the messages are encoded as JSON objects.

Upon input of a note for protection as well as the corresponding attribute-value pairs and parameters, the extension executes *PROTECT*. Attribute-value pairs can be defined in the interface (see Fig. 3). In order to publish a protected post, several attribute-value pairs can be chosen, and the cleartext of the post is entered (see Fig. 4).

The extension then encrypts the post and publishes the ciphertext within Facebook as a note, together with the encrypted shares and the attribute information. A PRNG provided by the Gibson Research Corporation[11] is used for generating a 256 bit key K. Encryption is done using the AES-256 implementation of the CryptoJS library[12] in CBC mode, and the SHA-256 implementation from the same library is used for hashing. K is split applying Shamir's secret sharing scheme, which leverages the fact that a polynomial of degree $t-1$ is uniquely identified by at least, but not less than t evaluations. We implement the finite field $GF(2^8)$ within our extension, and the secret is split into packets of 8 bits, each representing an element in $GF(2^8)$. For each of these packets, shares are computed and their concatenation then is encrypted using AES.

Upon access of a protected post, the user is presented with the requested attributes and can enter the values in corresponding text fields (see Fig. 6). A key is then constructed, and the post decrypted. In case the input values are entered correctly, the original post is displayed, otherwise the AES decryption yields the UTF-8 representation of the attempted decryption (cmp. Fig. 8 resp. Fig. 5). Attribute-value pairs both of the user and previously accessed protected posts are stored on the machine of the user only, using Mozilla's SQLite API.

6.2 Performance Evaluation

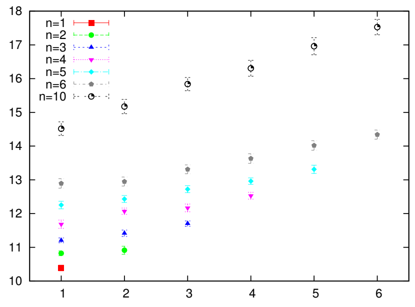
In order to test the applicability of our extension, we evaluate its performance on key generation as well as en- and decryption. We subsequently measure the time needed to access protected posts both for informed users and uninformed attackers, based on the analysis in Section 5.

Our measurements are performed on an average dual core PC with 3.00 GHz, 3.50 GB main memory and Windows 7 64 bit as operating system. Results are averaged over 100 runs and presented including their standard deviation. The duration of cryptographic operations depends on the total number of attributes n,

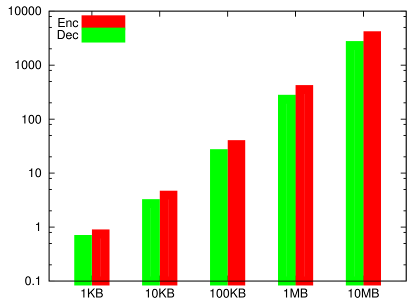
[10] `http://www.p2p.tu-darmstadt.de/research/PKA/`

[11] `https://www.grc.com/otg/uheprng.htm`

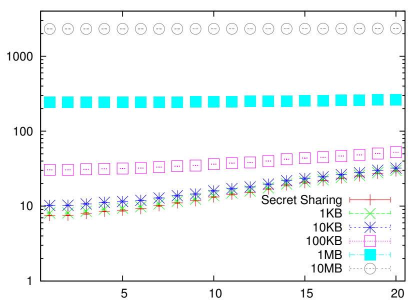
[12] `http://groups.google.com/group/crypto-js/topics`



(a) Key generation

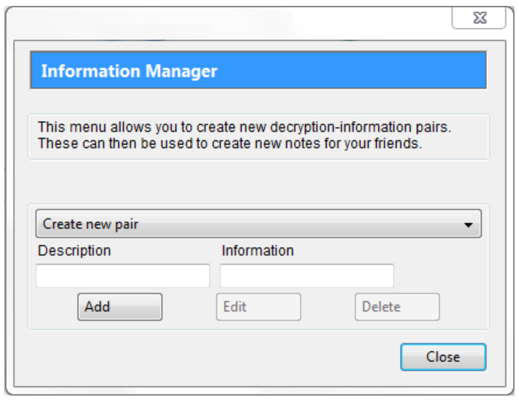


(b) En-/Decryption



(c) Total Time for Access

Figure 7: Time in ms needed for a) share generation and encryption for various t-of-n schemes, b) AES en-/decryption with given K for posts between 1 KB and 10 MB (log-log), c) total time for access for $n = 20$ and various thresholds t (log-log)



This menu allows you to create new decryption-information pairs. These can then be used to create new notes for your friends.

Figure 3: Dialog for managing attributes

the number of attributes needed to access the post t, as well as the post size.

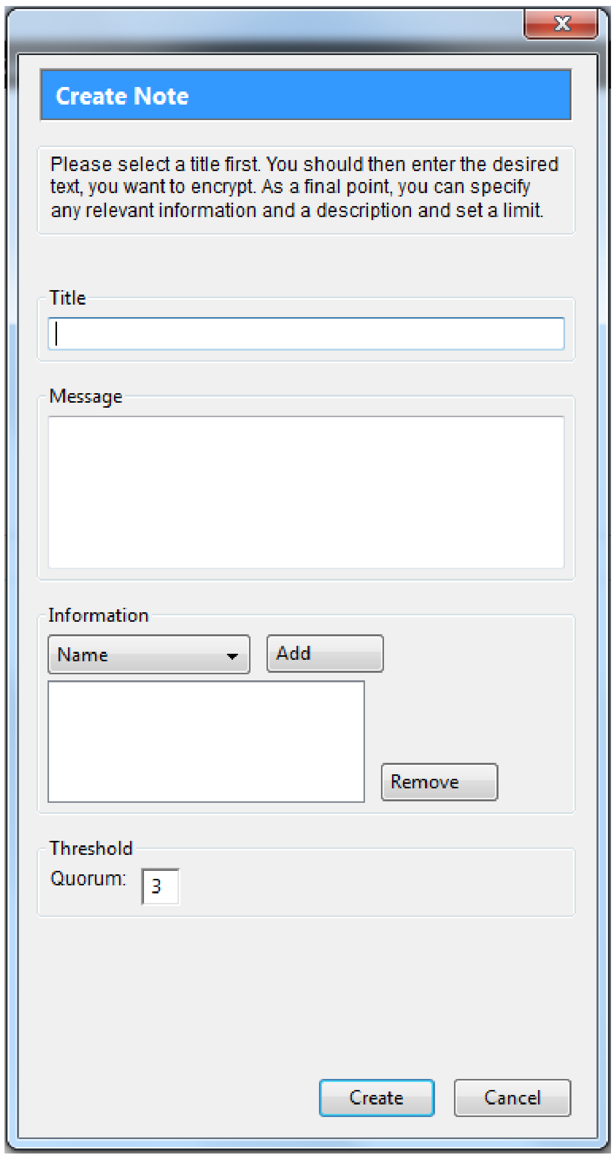
We vary the number of shares n between 1 and 20, whereas t is chosen between 1 and n. The following elementary operations are evaluated:

1. generation of keys (including share generation and encryption)
2. encryption and decryption of a post with known K
3. reconstruction of K from the encrypted shares.

The total time for access is the sum of key reconstruction and the decryption. We evaluate the en-/decryption cost for posts of sizes 1 KB to 10 MB, where the user input is chosen as random UTF-8 character strings.

Key generation.

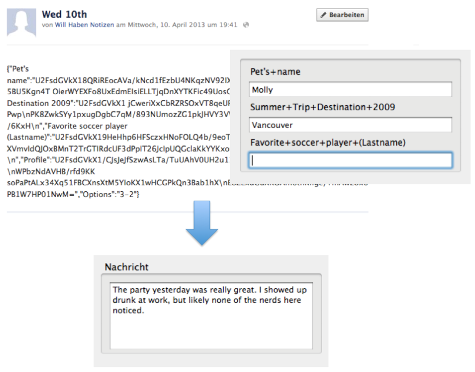
The key generation consists of the generation of the master key K, the generation of the $(t-1)$-degree polynomial p, the calculation of n shares and their encryption. Generating the master key takes 11.61 ms on average, with a standard deviation of 0.17 ms. Fig.



Please select a title first. You should then enter the desired text, you want to encrypt. As a final point, you can specify any relevant information and a description and set a limit.

Quorum: 3

Figure 4: Dialog for protecting a note


Wed 10th

von Will Haben Notizen am Mittwoch, 10. April 2013 um 19:41

{"Pet's
name":"U2FsdGVkX18QRiREocAVa/kNcd1fEzbU4NKqzNV92I
5BU5Kgn4T OierWYEXFo8UxEdmElsiELLTjqDnXYTKFic49Uos
Destination 2009":"U2FsdGVkX1 jCweriXxCbRZRSOxVT8qeU
Pwp\nPK8ZwkSYy1pxugDgbC7qM/893NUmozZG1pkJHVY3V
/6KxH\n","Favorite soccer player
(Lastname)":"U2FsdGVkX19HeHhp6HFSczxHNoFOLQ4b/9eoT
XVmvldQJOxBMnT2TrGTIRdcUF3dPplT26JclpUQGclaKkYYKxo
\n","Profile":"U2FsdGVkX1/CJsJeJfSzwAsLTa/TuUAhV0UH2u1
\nWPbzNdAVHB/rfd9KK
soPaPtALx34Xq51FBCXnsXtM5YloKX1wHCGPkQn3Bab1hX\n
PB1W7HP01NwM=","Options":"3-2"}

Pet's+name

Molly

Summer+Trip+Destination+2009

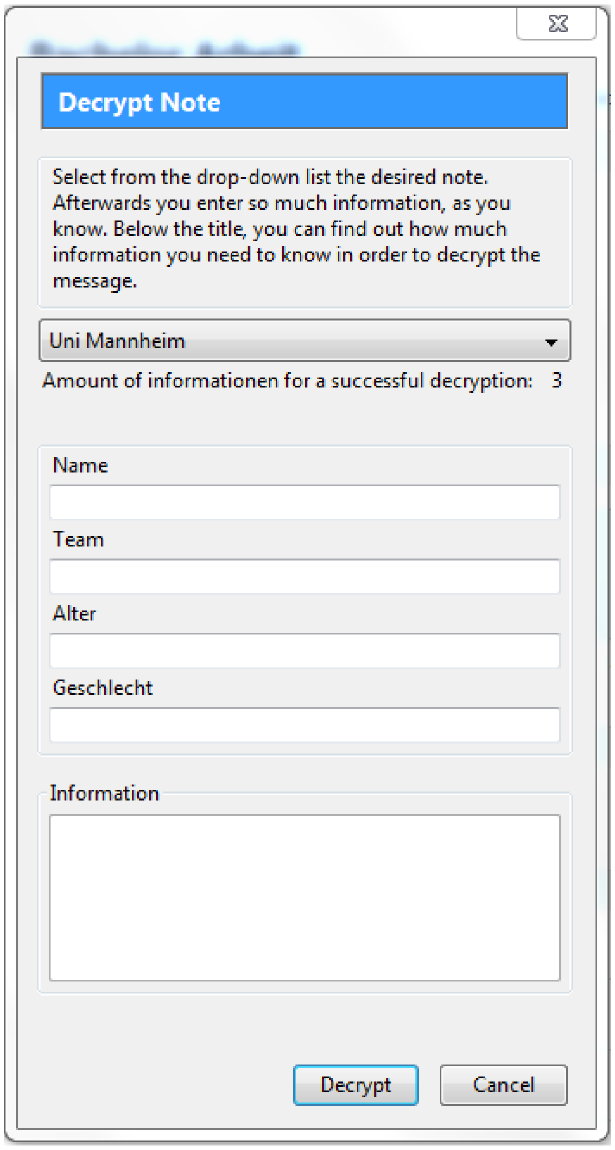
Vancouver

Favorite+soccer+player+(Lastname)

Nachricht

The party yesterday was really great. I showed up drunk at work, but likely none of the nerds here noticed.

Figure 5: A protected note, the attributes, and the accessed content displayed by the extension


Decrypt Note

Select from the drop-down list the desired note. Afterwards you enter so much information, as you know. Below the title, you can find out how much information you need to know in order to decrypt the message.

Uni Mannheim

Amount of informationen for a successful decryption: 3

Name

Team

Alter

Geschlecht

Information

Figure 6: Dialog for accessing a note

7a displays the measured times needed for share generation and encryption for $n \in \{1, 2, 3, 4, 5, 6, 10\}$ and $t \in \{1, 2, 3, 4, 5, 6\}$.

The time for generation and encryption indeed increases linearly with the number of attributes and the number of shares, but it remains below 15 ms for $t = 1$ in all experiments. Increasing the threshold to 5 shares, the time ranges from 13.67 ms ($n = 5$) to 16.47 ms ($n = 10$). In case of 20 shares, the average time for a 5-of-20 scheme is 25.78 ms, but increases to 42.86 ms when a threshold of 20 is used. In general, key generation and share encryption takes less than 45 ms, even in case of a scheme with 20 shares, which seems unlikely high for realistic cases.

The key generation, secret sharing, and share encryption hence takes too little time to even be perceived by the user.

En-/Decryption with known K.

We measure the duration of en-/decryption for posts sizes increasing by a factor 10 in each step. Fig. 7b indicates that the duration indeed increases linearly. The increase from 1 KB to 10 KB is slightly lower than in the other cases. The reason for this deviation is the fact that initial function call and processing the results demand a constant overhead independent of the size of the post. In case of pure textual posts the size is unlikely to exceed 100 KB[13], in which case the encryption takes well below 35 ms. Even in case of a hypothetical 10 MB post, including several photos for example, the encryption takes only slightly over 3s, which is on the order of time that downloading the same amount of data takes as well. Note that the median latency of accessing a webpage by a Desktop browser has been found to be 65 ms in 2012 [14], which is higher than the sum of key generation and encryption cost for realistic post sizes.

Key reconstruction.

Reconstructing K is expected to depend on t for decrypting the shares and performing the interpolation.

Fig. 7c displays the total time needed for accessing a post at various post sizes. These include the time for key reconstruction and decryption, and the former can be derived by subtracting the decryption time as measured above. The master key thus is reconstructed in less than 8 ms at $t \leq 3$, which we argue to be realistic in practice. The time for key reconstruction remains below 20 ms up to a threshold of 14, and even for $t = 20$ only roughly 30 ms are needed.

Total Time to Access.

[13] Note that Facebook notes are limited to a length of 65535 characters, yielding a maximum size of 128 KB UTF-8 encoded text.

[14] `http://www.webperformancetoday.com/2012/04/02/mobile-versus-desktop-latency/`

For *PKA* to be accepted, the total time for accessing a post is most critical. This is due to the facts that

1. *PROTECT* is only executed once per post, while *ACCESS* is executed by everyone interested in the post, and

2. an attacker accessing a protected post has to expect a delay of the number of needed trials times the average time for decryption.

Hence, accessing a post should be fast enough not to decrease the quality of service for a legitimate user, but make it unprofitable for an attacker to attempt gaining access.

Fig. 7c displays the total time access, consisting of key reconstruction and decryption, for various post sizes. It indicates that for small post sizes, the time for share retrieval increases noticeable with the threshold. A threshold of $t = 20$ takes around 10 times as long as threshold $t = 1$ (for a post size of 1 KB). The access time, however, generally remains on the order of hundreds of ms, and the delay of regular access is not noticeable. In case of large posts exceeding 1 MB, the delay of the key reconstruction is negligibly in comparison with the decryption, regardless of the secret sharing parameters.

Legitimate users may need several attempts on the first try until the correct values have been entered, of course. This is inevitable since a lot of values are bound to have synonyms. Assuming that such synonyms are few and that users carefully choose their values, we expect that a maximum of 10 combinations will have to be tested. With content amounting to sizes of several kilobytes, the total access time is still on the order of one second. The actual perceived overhead by the user hence clearly will be dominated by the time to manually select values for attributes, rather than the time needed by the processing of the extension. However, this cost is only applicable for the first trial of a user's keys. Assuming that attributes are frequently reused for protecting posts, all later posts are automatically decrypted by the extension, so that the latency is only that of the cryptographic operations, which is 31 ms for a 3-of-4 scheme and a post of 100 KB.

An uninformed attacker needs to consecutively guess attribute values, decrypt the shares, reconstruct candidates for K, and apply the decryption algorithm. Simplifying the estimation, we consider only the time needed to retrieve the shares, since the decryption depends on the size of the post. The results above show that in a 3-of-4 scheme, which seems realistic, a trial to retrieve the shares takes about 7.98 ms in average. Consulting the results from Section 5.2, we can estimate the time the attacker needs to successfully gain access to a post. Using the data sets meinVZ and XING, as described in Section 5.2, we consider value distributions

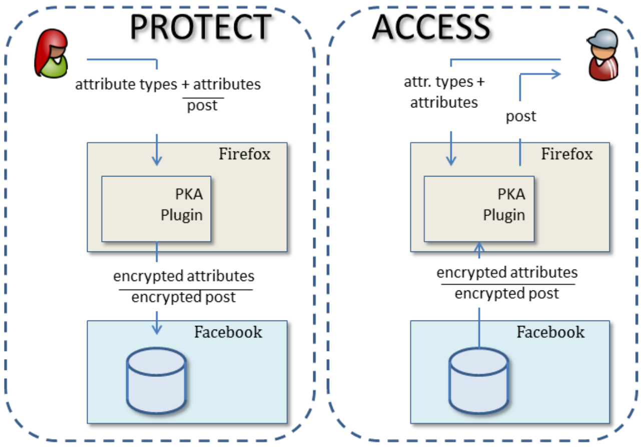


Figure 8: Interaction between user, extension and Facebook

derived from both OSNs as well as from the German statistical office (SB). Note that on average at least 54 days are needed to successfully access a protected post, for all considered basic and target distributions pairs. Since an attacker is not likely to test for all possible attributes, let us assume that he will try likely combinations for a minute, before turning to the next post. We see that, even in the case that the ground truth is known, the attacker is incapable of accessing about 90 % of the posts within one minute. The actual distribution, however, is not known to the attacker in reality. The best chance of the attacker is to assume that the distributions comply to corresponding official statistics. Considering the frequency distribution as found in the dataset of the German statistical office in more detail, however, only about 0.08% of the meinVZ and less than 1% of the XING profiles can be accessed in one minute.

The performance analysis shows that the protection and accessing time is barely noticeable for a legitimate user, but, assuming a sensible attribute selection, an attacker needs to invest days into accessing single posts. This represents a burden none of our considered adversaries - crawlers, third party applications, or the OSN provider - are willing to invest.

7. CONCLUSION

We have introduced PKA, a non-interactive scheme to protect public OSN data from crawlers. Though it is not the main application, protection against third-party applications, OSN providers, and arbitrary individuals can be achieved. User implicitly select their intended audience and level of security by their choice of attributes and values. Rather than replacing common access rules, PKA is an extension to the traditional layers of friends, friends-of-friends, and strangers, offering an additional, more fine-grained grouping of contacts. We have shown that PKA is secure under the assumption that the underlying cryptographic primitives - secret sharing scheme, symmetric encryption, and hashing - are secure. Furthermore, we estimated the number of trials needed to decrypt one post on the basis of data crawled from two OSNs. Even with rather unsuitable attribute choices, access is delayed on the order of minutes to days. Our Firefox extension, implementing PKA as proof of concept, supports the applicability of PKA, exhibiting negligible delays for legitimate access.

However, the current implementation focuses on providing the basic functionalities at a high level of security rather than usability. We plan on extending its functionalities to enable the encryption of non-textual content such as pictures. In addition, it should be easily possible to encrypt posts with complete previously used settings, whereas currently it is only possible to reuse attribute-value pairs. From a conceptional perspective, we currently evaluate how to use general access secret sharing [21] for weighted attributes.

8. REFERENCES

[1] D. Quercia, M. Kosinski, D. Stillwell, and J. Crowcroft. Our Twitter Profiles, Our Selves: Predicting Personality with Twitter. In *Socialcom*, 2011.

[2] David Malone and Kevin Maher. Investigating the distribution of password choices. In *WWW*, 2012.

[3] Arvind Narayanan and Vitaly Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In *CCS*, 2005.

[4] Amre Shakimov, Landon Cox, Alexander Varshavsky, and Ramon Caceres. Privacy, Cost, and Availability Trade-Offs in Decentralized OSNs. In *WOSN*, 2009.

[5] Antonio Cutillo, Refik Molva, and Thorsten Strufe. Safebook: Feasibility of transitive cooperation for privacy on a decentralized social network. In *WoWMoM*, 2009.

[6] Saikat Guha, Kevin Tang, and Paul Francis. Noyb: privacy in online social networks. In *WOSN*, 2008.

[7] Randy Baden, Adam Bender, Neil Spring, Bobby Bhattacharjee, and Daniel Starin. Persona: an online social network with user-defined privacy. In *SIGCOMM*, 2009.

[8] Sonia Jahid, Prateek Mittal, and Nikita Borisov. Easier: encryption-based access control in social networks with efficient revocation. In *ASIACCS*, 2011.

[9] Filipe Beato, Markulf Kohlweiss, and Karel Wouters. Scramble! your social network data. In *PETS*, 2011.

[10] Matthew M. Lucas and Nikita Borisov. flyByNight: Mitigating the Privacy Risks of Social Networking. In *SOUPS*, 2009.

[11] Amin Tootoonchian, Stefan Saroiu, Yashar Ganjali, and Alec Wolman. Lockr: Better Privacy for Social Networks. In *CoNEXT*, 2009.

[12] Wanying Luo, Qi Xie, and Urs Hengartner. FaceCloak: An Architecture for User Privacy on Social Networking Sites. In *CSE*, 2009.

[13] Sascha Fahl, Marian Harbach, Thomas Muders, and Matthew Smith. Trustsplit: usable confidentiality for social network messaging. In *HT*, 2012.

[14] Iulia Ion, Filipe Beato, Srdjan Capkun, Bart Preneel, and Marc Langheinrich. For some eyes only: protecting online information sharing. In *CODASPY*, 2013.

[15] Manuel Egele, Andreas Moser, Christopher Kruegel, and Engin Kirda. Pox: Protecting users from malicious facebook applications. *Elsevier ComCom*, 2012.

[16] Andreas Pashalidis, Nikos Mavrogiannopoulos, Xavier Ferrer, and Benat Bermejo Olaizola. For human eyes only: security and usability evaluation. In *WPES*, 2012.

[17] Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, FOCS, 1997.

[18] Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. All your contacts are belong to us. In *WWW*, 2009.

[19] Thorsten Strufe. Profile Popularity in a Business-oriented Online Social Network. In *EuroSys/SNS*, 2010.

[20] Adi Shamir. How to share a secret. *Comm. ACM*, 1979.

[21] M. Itoh, A. Saito, and T. Nishizeki. Secret sharing scheme realizing general access structure. In *IEEE Globecom*, 1987.