EASiER: Encryption-based Access Control in Social Networks with Efficient Revocation

Sonia Jahid
University of Illinois at Urbana-Champaign
sjahid2@illinois.edu

Prateek Mittal
University of Illinois at Urbana-Champaign
mittal2@illinois.edu

Nikita Borisov
University of Illinois at Urbana-Champaign
nikita@illinois.edu

ABSTRACT

A promising approach to mitigate the privacy risks in Online Social Networks (OSNs) is to shift access control enforcement from the OSN provider to the user by means of encryption. However, this creates the challenge of key management to support complex policies involved in OSNs and dynamic groups. To address this, we propose EASiER, an architecture that supports fine-grained access control policies and dynamic group membership by using attribute-based encryption. A key and novel feature of our architecture, however, is that it is possible to remove access from a user without issuing new keys to other users *or* re-encrypting existing ciphertexts. We achieve this by creating a proxy that participates in the decryption process and enforces revocation constraints. The proxy is minimally trusted and cannot decrypt ciphertexts or provide access to previously revoked users. We describe EASiER architecture and construction, provide performance evaluation, and prototype application of our approach on Facebook.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: Security and Protection; E.3 [**Data**]: Data Encryption

General Terms

Security, Algorithms

Keywords

Social Network, Access Control, Proxy, Revocation

1. INTRODUCTION

Online Social Networks (OSNs) such as Facebook, Myspace, and Orkut are becoming one of the most popular ways for users to interact online. However, this introduces a privacy risk, as the provider is an attractive attack target [6]. Insiders can also intentionally or accidentally release private information [4]. Several recent privacy compromises [13, 15] throw these issues into sharp focus. This motivates researchers to consider a paradigm shift, where instead of trusting social network operators and being dependent on them to enforce privacy, *users* are in control of who views their data, for example, via encryption [1, 5, 8, 9].

Fine-grained access control is a key challenge in this space; for example, Facebook and LiveJournal have rolled out mechanisms to specify access control policies for each post, as the data items are usually destined for a subset of friends, or groups. Persona [1] is a state-of-the-art design that proposes the use of attribute-based encryption (ABE) [2] to enable fine-grained access control. A user can create groups by assigning different attributes and keys to her social contacts, and then encrypt data such that only particular users having the desired set of attributes can decrypt it. This protects the information from unauthorized users on the OSN, third-party application developers, and above all the OSN itself.

However, groups are dynamic and therefore user attributes may change over time. This could be because of change in location, work environment, or the nature or strength of the relationship with a contact. Recent studies [10] have shown that the user interaction graph is much less dense than friendship graph, indicating that users interact most frequently with a small group of friends, further validating the need for fine-grained access control. Moreover, the churn rate for the interaction graph has been shown to be quite high [10], motivating the need for access control mechanisms to support *dynamic groups*. Persona and similar designs introduce significant overhead for group membership changes, especially when a contact is removed from a group: all other members of the group must receive a new key; additionally, all existing data items destined for that group must be re-encrypted. This does not scale when groups are large and dynamic, and when the volume of past data is high.

We propose EASiER, an architecture that enables users to set fine-grained access control policies even for dynamic groups. To handle group churn, we leverage ideas from the field of efficient revocation in group communications systems [12] and apply them to the ABE setting. Our design makes use of a minimally trusted proxy that handles revoked users and attributes. A user who revokes a contact or an attribute need not issue new keys to the rest of the group, nor re-encrypt data. We believe this feature is key for access control in OSNs, and would also be useful in any other

This work was supported in part by National Science Foundation Grant CNS 06–27671.

ASIACCS '11, March 22–24, 2011, Hong Kong, China.

context where ABE is used together with highly dynamic group membership. The proxy cannot decrypt by itself, and even if it were compromised, it cannot allow previously revoked users to decrypt either (unless the compromise crosses a proxy re-key operation). Therefore, a centralized OSN provider can act as a proxy without introducing significant privacy risks. The only assumption we hold is that the proxy is updated with a new key each time a revocation takes place and that it discards the old one.

We provide an overview of EASiER in Section 2 and a detailed description of our construction in Section 3. Section 4 contains our performance analysis, and finally, we describe related work in Section 5 and conclude in Section 6.

2. EASIER OVERVIEW

The primary goal of EASiER is to protect accidental or intentional information leak in OSN through encryption, specifically ABE, chosen for its expressiveness. Unlike traditional OSNs, which generally support one type of relationship such as friend, EASiER users define relationships by assigning attributes and keys to each other. To protect information, users encrypt different pieces of data such as profile information, wall posts, etc. with attribute policies. Only the contacts with keys having enough attributes to satisfy a policy can decrypt the data. For instance, in Figure 1, user A defines the attributes (*friend, colleague, neighbor*), generates keys k_1, k_2, and k_3 for the combination of attributes 'colleague', 'friend, neighbor', and 'colleague, neighbor' respectively, and assigns these keys to u_1, u_2, and u_3 respectively. She encrypts her data with the policy *'colleague or (friend and neighbor)'*.

Although ABE supports fine-grained policies, it leaves open the challenge of supporting dynamic groups, and in particular, revocation. This feature is a key functionality for OSNs, as relationships can and do change over time: A may want to cease its relationship with u_1 and u_2, and revoke the corresponding keys, which should prevent them from viewing A's data encrypted with any policy that their keys satisfy. Additionally, she may want to revoke the attribute 'neighbor' from k_3 assigned to u_3 and effect a corresponding change in access control. Traditional revocation approaches for ABE use frequent rekeying, and cannot block access to previously stored data without re-encrypting it.

The novel feature of EASiER is providing revocation in ABE by introducing a minimally trusted proxy. Since the proxy cannot decrypt the data directly, it may be implemented by a centralized service with minimal risk. Each user's proxy is assigned a secret proxy key with revocation information. The basic idea is, a social contact who wants to decrypt a data, takes a part of the ciphertext (CT) to the proxy. The proxy uses its key to transform CT into such a form that contains enough information that an unrevoked user can combine with his secret key mathematically, and successfully perform decryption, whereas a revoked user can not do so. Upon each key revocation, the user rekeys her proxy with the latest revocation information.

The ABE scheme used in EASiER is CP-ABE. We adopt the revocation scheme proposed by Naor and Pinkas [12] to support efficient revocation in ABE. The ciphertext in CP-ABE primarily consists of two parts: 1) the data blinded with some secret, and 2) one component for each attribute in the access structure used to encrypt the data. Without combining part 2 with the corresponding components

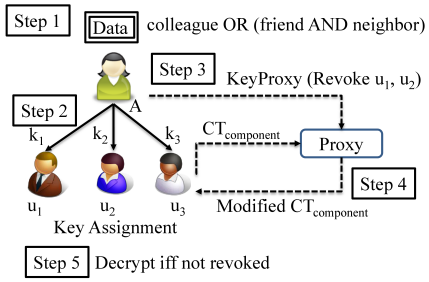


Figure 1: EASiER Architecture

in their attribute keys, users are unable to decrypt part 1. Our approach introduces an extra blinding to the attribute key components. The proxy key enables the unblinding of the components during a decryption for unrevoked users; revoked users, on the other hand, get no help from the proxy and are unable to decrypt any data, even if it was encrypted before the revocation event.

However, EASiER does not allow the proxy to decrypt the data since it does not have the attribute keys. A new proxy key, created each time a revocation takes place, prevents revoked users from colluding with the proxy or with each other to get the data. We argue that this is a desirable property: currently trusted contacts are not likely to crawl the entire set of social network data and store it for later use, but former friends or colleagues might try to abuse their former status by accessing past data. Figure 1 summarizes the whole architecture.

3. CONSTRUCTION

3.1 Assumptions and Basics

Before going into details of the construction, we present some basic mathematical assumptions, and some details of CP-ABE and the revocation scheme on which EASiER is built:

Bilinear Pairing: Let $\mathbb{G}_0$, $\mathbb{G}_1$, and $\mathbb{G}_2$ be multiplicative cyclic groups of prime order p, and e a map $(\mathbb{G}_0 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2)$. If $\forall u \in \mathbb{G}_0, v \in \mathbb{G}_1$ and $a, b \in Z_p$, $e(u^a, v^b) = e(u, v)^{ab}$ and $e(g_0, g_1) \neq 1$, then e is a bilinear pairing. If $\mathbb{G}_0 = \mathbb{G}_1$, it is a symmetric pairing, otherwise the pairing is asymmetric.

Secret Sharing: In Shamir's secret sharing scheme [14], a secret S in some field F is shared among n parties by creating a random polynomial $P \in F[x]$ of degree t such that $P(0) = S$. The i-th party gets the share $(i, P(i))$. Given any $t+1$ shares $P(x_0), \ldots, P(x_t)$, it is possible to recover $P(0)$ using Lagrange interpolation:

$$P(0) = \sum_{i=0}^{t} \lambda_i P(x_i), \quad \text{where } \lambda_i = \prod_{j \neq i} \frac{x_j}{(x_j - x_i)}$$

CP-ABE: We summarize the relevant CP-ABE algorithms; for more details, refer to [2].

- **Setup:** The key authority (KA) generates a public key PK, and a master secret key MK, for random $\alpha, \beta \in Z_p, \mathbb{G}_0 = \langle g \rangle$:

$$PK = \mathbb{G}_0, g, h = g^\beta, e(g,g)^\alpha, MK = (\beta, g^\alpha)$$

- **Encrypt(PK, M, τ):** It takes PK, the data M, and a policy τ represented as a tree access structure. Let random $s \in Z_p$ be the secret at the root of the tree, q_x

be the polynomial of degree $d_x = k_x - 1$ at the node x where k_x is the threshold of x, and Y be the set of leaf nodes in τ. The ciphertext CT is:

$$CT = \big(\tau, \tilde{C} = Me(g,g)^{\alpha s}, C = h^s, \\ \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(\text{att}(y))^{q_y(0)}\big)$$

Here, $H : \{0,1\}^* \to \mathbb{G}_0$ is a hash function that maps a string attribute to a random element of $\mathbb{G}_0$.

- **KeyGen(MK, S):** It takes MK, and a set of attributes S and generates the secret key SK corresponding to S for random $r, r_j \in Z_p$:

$$SK = \big(D = g^{(\alpha+r)/\beta}, \\ \forall j \in S : D_j = g^r H(j)^{r_j}, D'_j = g^{r_j}\big)$$

- **Decrypt(CT, SK)**: A recursive algorithm *DecryptNode* pairs D_i and D'_i (from SK) with C_x and C'_x (from CT), respectively, and returns $e(g,g)^{rq_x(0)}$ for each leaf node x in the τ in CT, iff $i = \text{att}(x)$ for some $i \in S$, the set of attributes assigned to SK. At each non-leaf node x, Lagrange interpolation is used on at least k_x such $e(g,g)^{rq_{z_j}(0)}$ from its children $\{z_j\}$, to calculate $e(g,g)^{rq_x(0)}$. Let $A = e(g,g)^{rq_R(0)} = e(g,g)^{rs}$. $\tilde{C}, C, D$ and A are used in bilinear mapping to retrieve M.

Revocation Scheme of Naor and Pinkas: This scheme consists of 2 phases:

- **Initialization:** The group controller generates a random polynomial P of degree t over Z_p. It sends a personal key $\langle I_u, P(I_u)\rangle$ to each user u with identity I_u. This process is performed only once for all future revocations.
- **Revocation:** The group controller learns the identities of t users $I_{u1}, \ldots, I_{ut}$ to revoke. It then chooses a random r, and sets the new key to be $g^{rP(0)}$, which would be unknown to revoked users. It broadcasts the message $g^r, \langle I_{u1}, g^{rP(I_{u1})}\rangle, \ldots, \langle I_{ut}, g^{rP(I_{ut})}\rangle$ encrypted with the current group key. Each non revoked user can compute $g^{rP(I_u)}$ and combine it with the broadcast values to obtain $g^{rP(0)}$ using Lagrange interpolation. Further details can be found in [12].

3.2 Key and Attribute Revocation in EASiER

In this section we describe how to revoke keys and attributes from parties in EASiER. This construction allows revocation of up to t users at a time for a complete key revocation, or t_y users for each attribute y in case of attribute revocation, since it is based on the scheme in [12] described before. At first we describe key revocation.

- **Setup:** Our scheme requires an asymmetric pairing $e : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_2$ where there is no efficient isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_0$. Each OSN user acts as her own KA, and generates PK and MK for random polynomial P of degree t (the maximum number of revoked users) over Z_p, generators g_0 and g_1 of $\mathbb{G}_0$ and $\mathbb{G}_1$, respectively, and $\alpha, \beta \in Z_p$.

$$PK = \mathbb{G}_0, \mathbb{G}_1, g_0, g_1, h = g_0^{\beta}, e(g_0,g_1)^{\alpha}, MK = \beta, g_1^{\alpha}, P$$

- **KeyGen(MK, S):** It outputs the secret key corresponding to the set of attributes S, blinded by $P(0)$, the secret to be used after revocation. We introduce an extra component—D''_j—that in addition to attribute information contains user information. Without loss of generality, we assume user u_k receives this key.

$$SK = (D, \forall j \in S : \langle D_j, D'_j, D''_j\rangle), \text{where} \\ D = g_1^{(\alpha+r)/\beta}, D_j = g_1^r \cdot H(j)^{r_j P(0)} = g_1^{r + h_j r_j P(0)}, \\ D'_j = g_0^{r_j}, D''_j = (D'_j)^{P(u_k)} = g_0^{r_j P(u_k)}$$

where $H : \{0,1\}^* \to \mathbb{G}_1$ and $h_j = \log_{g_1} H(j)$ (used for notational convenience only).

- **Encrypt(PK, M, τ):** This algorithm is exactly the same as in CP-ABE, adjusted for the use of an asymmetric pairing:

$$CT = (\tau, \tilde{C} = Me(g_0,g_1)^{\alpha s}, C = h^s = g_0^{\beta s}, \\ \forall y \in Y : C_y = g_0^{q_y(0)},\ C'_y = H(att(y))^{q_y(0)} = g_1^{h_y q_y(0)})$$

- **ProxyRekey(PK, MK, RL):** Whenever the owner wants to revoke keys from social contacts, she creates a list of revoked users RL with their identities u_i, $i \in \{1, \ldots, t\}$, and evaluates the corresponding $P(u_i)$ using MK. She gives the proxy key PXK to the proxy. In case of fewer than t revocations, the owner generates random $\langle x, P(x)\rangle$ such that x does not correspond to any user's identity to pad PXK to length t.

$$PXK = \forall u_i \in RL : \langle u_i, P(u_i)\rangle$$

- **Convert(PXK, $\forall y \in Y : C_y$, u_k):** The proxy uses its key, and the decryptor's identity u_k to calculate C''_y as follows:

$$\lambda_i = \frac{u_k}{u_k - u_i} \prod_{j \neq i} \frac{u_j}{(u_j - u_i)}, \forall i, j \in \{1, \ldots, t\},\ k \notin \{1, \ldots, t\}$$

$$\forall y \in Y : C''_y = (C'_y)^{\sum_{i=1}^{t} \lambda_i P(u_i)} = g_1^{h_y q_y(0) \sum_{i=1}^{t} \lambda_i P(u_i)}$$

Since the user's secret key SK is blinded by $P(0)$, she needs C''_y in addition to C_y and C'_y for decryption.The proxy also calculates and gives λ_k to the user u_k.

- **Decrypt(CT, SK)**: The decryption steps involve one extra pairing than basic CP-ABE at each leaf node of the policy. For each leaf node x, with $i = \text{att}(x)$, if $i \in S$, where S is the set of attributes for SK, then,

$$\begin{aligned} &DecryptNode(CT, SK, x) \\ &= \frac{e(C_x, D_i)}{e(D''_i, C'_x)^{\lambda_k} e(D'_i, C''_x)} \\ &= \frac{e(g_0,g_1)^{rq_x(0) + h_i r_i P(0) q_x(0)}}{e(g_0,g_1)^{r_i h_i q_x(0) \lambda_k P(u_k)} e(g_0,g_1)^{r_i h_i q_x(0) \sum_{j=1}^{t} \lambda_j P(u_j)}} \\ &= \frac{e(g_0,g_1)^{rq_x(0) + h_i r_i P(0) q_x(0)}}{e(g_0,g_1)^{r_i h_i q_x(0) (\sum_{j=1}^{t} \lambda_j P(u_j) + \lambda_k P(u_k))}} \\ &= e(g_0,g_1)^{rq_x(0)} \end{aligned}$$

Otherwise *DecryptNode* returns $\perp$. The rest of the decryption is the same as CP-ABE. For each child z of a non-leaf node x, it calculates $F_z = e(g_0,g_1)^{rq_z(0)}$. Let S_x be a threshold-sized arbitrary set of children of x, such that $F_z \neq \perp$. Then interpolation and pairings are used to calculate $e(g_0,g_1)^{\alpha s}$, and hence retrieve M.

The construction to revoke one or more attributes from a key is almost similar to the one just described. From

high level, the primary difference is that, instead of using a single polynomial in MK, a KA uses a different polynomial for each of the attributes she defines. Each of the attribute-components in the secret key SK is blinded by the corresponding polynomial, and the PXK contains a list of revoked users and the secrets corresponding to the revoked attributes of each user. The algorithms work as before. Let Y' be the set of attributes that a KA defines, Y be the set of revoked attributes where $Y \subseteq Y'$, and t_y be the number of users from whom attribute $y \in Y$ is revoked.

$$MK = \beta, g_1^{\alpha}, \forall y \in Y' \; P_y(0)$$
$$SK = \left(D, D_j = g_1^r \cdot H(j)^{r_j P_j(0)}, D_j'' = (D_j')^{P_j(u_k)}\right)$$
$$PXK = \forall y \in Y, \; \forall i \in t_y : \; \{\langle u_i, P_y(u_i)\rangle\}$$

4. IMPLEMENTATION AND EXPERIMENTAL EVALUATION

We implemented the constructions in EASiER, as described in Section 3. Our implementation involves introducing new components as well as modifying different parts of the CP-ABE toolkit[1]. The current implementation supports complete key revocation. Similar techniques can be applied to modify it to perform attribute revocation.

The implementation uses MNT curves [11] with a 159-bit base field. All the experiments were carried out on a 2.40 GHz Intel Core 2 Duo, 3 GB memory, and running Ubuntu 8.10. We also implemented a Facebook application to provide the functionality on a social network. The prototype Facebook application can be found at `http://apps.facebook.com/myeasier`.

4.1 Performance Analysis

We provide some information on the performance evaluation of EASiER, and compare it with CP-ABE. Though CP-ABE implementation uses symmetric pairing, we use asymmetric pairing for both EASiER and CP-ABE in our implementation. This provides security by preventing key and ciphertext components exchange. The results are shown in Figure 2.

Key generation time is linear with number of attributes both in CP-ABE and EASiER. Since it does an extra exponentiation, and generates an extra component for each attribute in EASiER, the result is justified. To test encryption and decryption, we randomly generated 10 different policies for each of the desired number of leaves $(1, 5, 10, \ldots 100)$. Encryption (not shown in the Figure) is also linear with respect to the number of leaf nodes in the policy. Since no change was made to CP-ABE encryption, both take the same amount of time. Decryption (Figure 2b shown with 95% confidence interval) depends on the policy used in encryption and the attributes involved. We generated a decryption key with 100 attributes that satisfies all the policies used. The line marked EASiER-naive shows the decryption time when no optimization was used. Recursive *DecryptNode*, and arbitrary number of leaves were used to satisfy the threshold gates in the policies. The lines marked EASiER, and CP-ABE show the results when an optimization implemented in CP-ABE was used to ensure that the minimum number of leaves were used in *DecryptNode*. The required time is still below 1 second though recursive *DecryptNode* was used. We expect better results with further optimization.

EASiER involves two extra costs before decryption: re-keying the proxy and converting the ciphertext components specific to the leaves in the policy. We perform an optimization by allowing the proxy to pre-calculate a portion of the λ_i's in **ProxyRekey**. The re-keying results (not shown in Figure) show that, even for 500 revoked users, the time required is about 1.4 seconds. This should be compared with the time required to rekey the rest of a group, i.e., generate a new key for everyone, when even one person in the group is revoked.

$$\lambda_i' = \prod_{u_i, u_j \in RL, i \neq j} \frac{u_j}{(u_j - u_i)}, \text{ and } l_i' = \lambda_i' P(u_i)$$

Conversion primarily involves one exponentiation for each of the leaf-specific ciphertext component. It also calculates λ_k for the requester u_k, and completes the $\lambda_i s$ for each of the revoked users. With the optimization, the proxy needs to do 1 multiplication per revoked user to calculate λ_i. It works as follows:

$$l_i = l_i' \frac{u_k}{(u_k - u_i)} = \lambda_i' \frac{u_k}{(u_k - u_i)} P(u_i) = \lambda_i P(u_i), \quad \forall \; u_i \in RL, u_k \notin RL$$

The time to compute the exponentiations dominates the time to do t multiplications, hence the results are essentially linear in the number of leaf nodes. Figure 2c shows the conversion time for 500 revoked users. We expect the proxy to be more powerful in terms of computing, and hence rekeying, and conversion should be faster in practice. A user requesting decryption only faces the conversion time shown in Figure 2c along with the decryption time mentioned earlier.

Table 1: Component Size

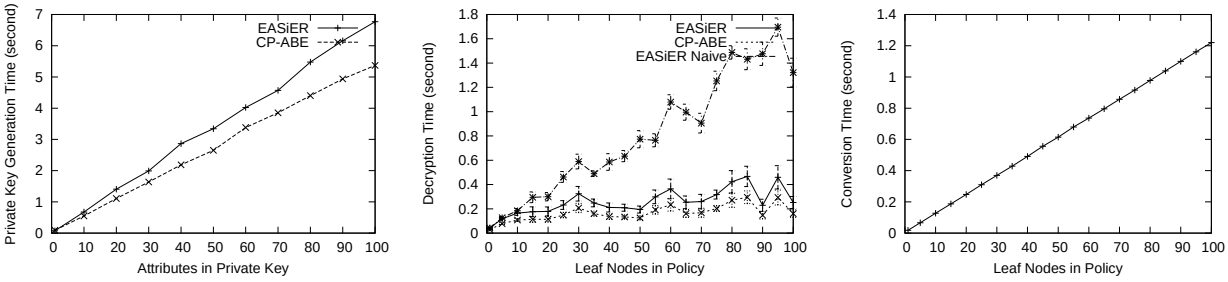
Component	EASiER (bytes)	CP-ABE (bytes)
Public Key	1316	1316
Master Key	$152 + (t+1)24$	148
Private Key	$128 + (a+212)n$	$128 + (a+168)n$
Ciphertext	$168 + 8i + (176+a)l$	$168 + 8i + (176+a)l$
Proxy Key	$24t$	NA
C_y''	$124l$	NA

4.2 Component Size and Communication Overhead

Table 1 shows the sizes of the components involved in the system, calculated based on group members and elements they consist of. t is the degree of polynomial, n is the number of attributes in private key, and a is the string length of an attribute. Elements from $\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_2$, and Z_p require $44, 124, 124$, and 24 bytes respectively to represent. Public Key includes a string describing the pairing used (980 bytes).

Users communicate with the proxy for conversion by sending C_y', and receiving C_y''. These are represented using elements from $\mathbb{G}_1$. This requires 124 bytes to represent (120 for the actual data, and 4 for the variable size). Hence, conversion of a ciphertext with l leaf nodes in the policy will need to transfer $124l$ bytes each way. The user also sends u_k, and receives λ_k back. These are represented using Z_p which requires 24 bytes. Therefore, we conclude that the communication overhead is reasonable for OSN users.

[1] `http://acsc.cs.utexas.edu/cpabe/`



(a) Key Generation (b) Decryption (c) Conversion in EASiER

Figure 2: EASiER Performance Analysis

5. RELATED WORK

In most of the existing social network privacy architectures settings access control is performed via encryption, but none of the schemes focus on the issue of efficient user or attribute revocation. The existing revocation schemes have their limitations too.

Persona [1] is the state-of-the-art decentralized architecture for social network privacy. EASiER is based on Persona, but it also provides an efficient mechanism for user/attribute revocation, thereby avoiding the overhead of re-keying with group members and re-encryption of old data. In flyByNight [8], users encrypt sensitive messages using Javascript on the client side, such that the sensitive data cannot be viewed by the OSN servers. flyByNight relies on the OSN provider for key management and is vulnerable to active attacks by the provider. The key feature of NOYB [5], an architecture for OSN is a general cipher and encoding scheme that preserves the semantic properties of data such that it can be processed by the social network provider *oblivious* to encryption. FaceCloak [9] has a similar goal, but opts to store the encrypted data on a third-party server, with fake data stored at the OSN provider.

Liang et al. [7] propose a proxy-based re-encryption scheme which can transform a ciphertext encrypted with one specific policy to one encrypted with another. It is infeasible to apply their techniques to our problem, as a separate proxy key would be required for every possible encryption policy. Boldyreva et al. [3] propose a revocation scheme for key-policy ABE where a single key update is broadcast to all users, but a revoked user maintains access to previously encrypted data. Finally, Yu et al. [16] define an architecture for revocation in CP-ABE that requires the re-issuing of keys and re-encryption of existing data, but delegates this task to a proxy server, rather than the authority; it also does not consider the issue of proxy server compromise.

6. CONCLUSION

We present an access control architecture for OSNs, named EASiER that supports efficient revocation in ABE. We achieve this revocation scheme by introducing a minimally trusted proxy, leveraging ideas from a group communication scheme, and combining it with ABE. Although we showed our approach in an OSN setting, it can be applied to any context where ABE is implemented. We implemented the scheme and compared it with Bethencourt et al.'s CP-ABE. Our results show that EASiER is scalable in terms of computation and communication for OSNs; accordingly, we have built a prototype application in the Facebook OSN to provide such encryption. Our future plans include an investigation into alternate CP-ABE constructs to be used with EASiER in order to achieve stronger security guarantees.

7. REFERENCES

[1] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin. Persona: An online social network with user-defined privacy. In *SIGCOMM*, 2009.

[2] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE S & P*, 2007.

[3] A. Boldyreva, V. Goyal, and V. Kumar. Identity-based encryption with efficient revocation. In *CCS*, 2008.

[4] R. Gross and A. Acquisti. Information revelation and privacy in online social networks (the Facebook case). In *WPES*, 2005.

[5] S. Guha, K. Tang, and P. Francis. NOYB: Privacy in online social networks. In *WOSN*, 2008.

[6] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer. Social phishing. *Commun. ACM*, 50(10):94–100, 2007.

[7] X. Liang, Z. Cao, H. Lin, and J. Shao. Attribute based proxy re-encryption with delegating capabilities. In *ASIACCS*, 2009.

[8] M. M. Lucas and N. Borisov. flyByNight: Mitigating the privacy risks of social networking. In *WPES*, 2008.

[9] W. Luo, Q. Xie, and U. Hengartner. FaceCloak: An architecture for user privacy on social networking sites. In *PASSAT*, 2009.

[10] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *IMC*, 2007.

[11] A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Trans. Fundamentals*, E84-A(5):1234–1243, 2001.

[12] M. Naor and B. Pinkas. Efficient trace and revoke schemes. In *FC*, 2001.

[13] M. O'Connor. Facebook revealed private email addresses last night. `http://gawker.com/5505967/facebook-revealed-private-email-addresses-last-night`, Mar.31 2010.

[14] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[15] P. Wong. Conversations about the Internet #5: Anonymous Facebook employee. The Rumpus, 2010.

[16] S. Yu, C. Wang, K. Ren, and W. Lou. Attribute based data sharing with attribute revocation. In *ASIACCS*, 2010.