Lockr: Better Privacy for Social Networks

Amin Tootoonchian
Computer Science
University of Toronto

Stefan Saroiu
Microsoft Research
Redmond, WA

Yashar Ganjali
Computer Science
University of Toronto

Alec Wolman
Microsoft Research
Redmond, WA

ABSTRACT

Today's online social networking (OSN) sites do little to protect the privacy of their users' social networking information. Given the highly sensitive nature of the information these sites store, it is understandable that many users feel victimized and disempowered by OSN providers' terms of service. This paper presents Lockr, a system that improves the privacy of centralized and decentralized online content sharing systems. Lockr offers three significant privacy benefits to OSN users. First, it separates social networking content from all other functionality that OSNs provide. This decoupling lets users control their own social information: they can decide which OSN provider should store it, which third parties should have access to it, or they can even choose to manage it themselves. Such flexibility better accommodates OSN users' privacy needs and preferences. Second, Lockr ensures that digitally signed social relationships needed to access social data cannot be re-used by the OSN for unintended purposes. This feature drastically reduces the value to others of social content that users entrust to OSN providers. Finally, Lockr enables message encryption using a social relationship key. This key lets two strangers with a common friend verify their relationship without exposing it to others, a common privacy threat when sharing data in a decentralized scenario.

This paper relates Lockr's design and implementation and shows how we integrate it with Flickr, a centralized OSN, and BitTorrent, a decentralized one. Our implementation demonstrates Lockr's critical primary benefits for privacy as well as its secondary benefits for simplifying site management and accelerating content delivery. These benefits were achieved with negligible performance cost and overhead.

Categories and Subject Descriptors

C.2.0 [**General**]: Security and protection

CoNEXT '09 , December 1-4, 2009, Rome, Italy.

General Terms

Security, Design, Experimentation

Keywords

social networks, privacy, social attestation, witness hiding

1. INTRODUCTION

Most Internet users lack awareness of the privacy risks posed by storing their social information in online social networking (OSN) sites, such as Facebook or LinkedIn. Their outrage at discovering that they cannot permanently delete information about friends and family members from these sites, for example, or that they cannot prevent their information from being divulged or sold to advertisers [24], has been amply documented by mainstream media [1, 37, 39]. It has also led to FTC complaints by privacy watchdogs [27] and to the rise of grass-roots movements to prevent further erosion of privacy when OSNs modify their Terms-of-Service [38]. Further, risks to OSN data from security attacks or accidental disclosure are significant [3]. Despite these considerable threats, today's OSN users have no choice but to "trust" OSNs with the identities of their family members, friends, and colleagues.

OSNs have thus far done little to inform users about the privacy implications of using their sites or to offer simple and intuitive privacy controls. In fact, many sites view ownership of social networking information as a key ingredient of their revenue streams. They therefore have little incentive to make it possible for users to control it. Despite this current reticence, we believe that government, media, and public pressure will force the issue and redress the power imbalance. We also believe that OSNs with compelling functionality will continue to remain popular, and thus financially viable, once they have ceded data control to their users.

Compounding these privacy risks are the management headaches users face when interacting with their OSNs. To access their friends' private content, today's OSN users have two choices: (1) they must register with all the sites that their friends use, or (2) they must wait to receive a secret URL where they can go view the content. With multiple registrations, OSN users must duplicate portions of their social network information on many different OSN sites. With secret URLs, users must send the correct URLs for each OSN site via out-of-band mechanisms (e.g., e-mail) to their friends. Both are tedious tasks that further raise the risk of data loss or mismanagement.

This paper takes an initial step toward addressing the pri-

vacy problems that have plagued today's OSNs. We present Lockr, a system that offers three key privacy benefits for OSN users.

1. *Social attestations* to decouple social networking information from other OSN functionality. Lockr users need not provide a full copy of their social network to each online site they use to host their personal content. Instead, they exchange *social attestations*: application-independent small pieces of digitally signed meta-data issued by one person to another that encapsulate social relationships. The recipient of a social attestation can use it to prove the social relationship to any online system (e.g., Alice presents an attestation from Bob that says: "Bob says Alice is my friend"). Although inspired by capabilities [25], attestations do not include access rights. When making content available, owners associate a *social access control list* with their content to restrict access to only those people who have a specific social relationship to them. They exchange attestations *once*, yet they can use them repeatedly to access their friends' online content without the need to register with many OSNs, to maintain many copies of their social networks, or to exchange secret URLs.

2. *The WHPOK protocol* to protect social information from disclosure. Lockr prevents an OSN from reusing the digitally signed social relationships revealed when users access their data by applying to the attestation verification mechanism a variant of zero-knowledge protocols, called a *witness hiding proof of knowledge (WHPOK) [20] protocol*. This lets users prove the existence of a social relationship without providing a verifiable copy of the attestation that Web sites could store and later reuse. Without a signature, attestations have little value to others: they are just pieces of XML text that could easily have been fabricated by the Web site.

3. *Relationship keys* to solve "social deadlock" and *one-way hash chains* to protect data from privacy abuse via expired attestations. Sharing information in decentralized scenarios, such as peer-to-peer [35, 7, 11] scenarios, give rise to the following problem: two peers would like to exchange content with each other only if they share a specific common friend, yet neither wants to reveal this social relationship to a random peer that does not share this common friend. This creates a form of "deadlock", where each peer is waiting on the other one to reveal its relationship first. Lockr lets peers encrypt their communication to each other using a key specific to a relationship. In this way, no receiving peer can decrypt a message unless it has the same social relationship as the message's sender. Lockr thus lets two random strangers who share a friend verify their common social connection without a loss of privacy. In addition, Lockr uses one-way hash chains to ensure that relationship keys are invalidated once an attestation expires to prevent privacy abuse.

Lockr works with both centralized Web sites and decentralized P2P systems. We demonstrate its generality by describing implementations that integrate Lockr with the Flickr photo sharing Web site and the BitTorrent P2P file sharing system. We use the WHPOK protocol in our implementation of Lockr for BitTorrent, but we leave an implementation of WHPOK in Lockr for Flickr for future work. However, we were able to integrate Lockr with Flickr even without Flickr's cooperation. In fact, it is possible to deploy Lockr on some of today's OSN sites absent server-side support. To accomplish this, Lockr relies on a proxy server that compensates for missing server-side functionality.

We continue this paper with a discussion of Lockr's benefits. Section 3 then describes Lockr's design components and design goals, while Section 4 presents implementation details of integrating Lockr with BitTorrent and Flickr. We then evaluate Lockr's performance (Section 5) and discuss how Lockr relates to prior work (Section 6) before concluding our discussion (Section 7).

2. BENEFITS OF THE LOCKR ARCHITECTURE

Today's OSNs have adopted a centralized, single-site design in which users join a site to create social relationships with other registered users. The OSN's role is two-fold: (1) to manage social information, and (2) to offer various social applications, including content delivery and sharing. Unfortunately, OSNs have tightly coupled these two roles. Thus, users can neither manage their social network outside the OSN's site nor use their network with other third-party applications.

Our premise is that *this tight coupling of social information and functionality into one centralized solution has led to many of the privacy challenges faced by today's OSN users.* Working from this premise, the following section explores both Lockr's primary benefits – enhancing privacy – and secondary benefits of simplifying site management and accelerating content delivery.

2.1 Enhancing Privacy

To offer an adequate level of privacy, OSNs must give users a choice about where to store their social information. Users should be able to choose a single trusted OSN provider – one with acceptable privacy policies – yet use the applications offered by any other OSN. If no OSN is adequate, users should be able to manage their social information themselves. Otherwise, they will continue to be surprised and blind-sided by OSN privacy policies.

Lockr decouples OSN social information from functionality. It treats social information as a resource managed and exchanged by people via any mechanism they find convenient, such as OSN sites, e-mail, business cards, or Bluetooth-equipped phones. For example, we show how Facebook could adopt Lockr; we implemented an address book as a Facebook application to let users exchange social attestations in Facebook and in other applications.

Lockr's decoupled architecture offers these privacy benefits:

1. Improved confidentiality of OSN content. OSN users need not reveal a full copy of their social network to every OSN site they use. Because OSNs no longer store copies of their users' social networks, the chances of mismanagement or accidental disclosure of social networking information are drastically reduced. However, Lockr lets OSNs continue to serve content, such as photos and videos, as well as host third-party social applications.

2. Simplified OSN site registration. Lockr users need not register with a site to access their friends' content. Instead, Lockr users must prove only that they own a digitally signed attestation from their friend.

3. Enhanced confidentiality upon access to OSN. Lockr's privacy-preserving protocols let users prove the existence of a digitally signed attestation to an OSN site without actually having to reveal the digital signature as part of the verification process. This reduces the "resale" value of the content they must share with OSN providers.

2.2 Simplifying Site Management

To share online personal content, today's OSN users must divide their content according to its type (e.g., photos, videos, and Web bookmarks) and place each of these types on different sites. For example, they must place photographs on Flickr, videos on YouTube, and Web bookmarks on del.icio.us. The content management mechanisms offered by these Web sites often require all participants to be registered with the site in question to view their friends' private content or to exchange secret URLs pointing to the content's location. Moving content to a new system requires inviting all of one's friends to that system; those friends must have to register before they can use the new OSN site. In addition, an individual's login ID may differ across the various content sharing Web sites, further complicating the process of giving access to a specific social group. Most OSN providers do not assist with the process of reconciling one's social network across multiple content sharing sites.

In fact, OSN sites employ management mechanisms reminiscent of those that operating systems and databases offered in the past. Users must create an identity with each online system and then create groups by enumerating other identities that are allowed access to their content. Furthermore, the details differ from site to site in terms of the capabilities and semantics of the provided access control mechanisms. This inevitably leads to confusion and errors in specifying appropriate access control rules and in maintaining copies of one's social network. As a result, people often compromise on privacy and choose to make all their personal content publicly accessible simply to avoid such complications.

Lockr's decoupling eliminates the burden on users of maintaining several up-to-date copies of social networks, performing user-id reconciliation across sites, and familiarizing themselves with the varied access control mechanisms provided by each site. Instead, they can express the same simple access control policy on any OSN site that stores their content. For example, to restrict access to a picture to friends only, a user just creates a simple policy that lists the relationship name (i.e., "friend") in the access control list. The access control list format is specific to the attestations users exchange with each other, and thus it remains the same on any OSN site.

2.2.1 Improving Content Delivery Performance

OSNs require their users to upload any content they want to share with their friends. To do so, users must convert their content into the format supported by the OSN site, often a format with a lower quality that minimizes storage requirements. When a piece of content becomes popular, OSN sites must rely on expensive content distribution mechanisms, such as CDNs, to handle the intense bandwidth requirements. They do not currently use free content distribution systems, such as BitTorrent or other P2P systems. Doing so would require peers to access the centralized OSN site to obtain their identities and social information. The unnecessary coupling of social information and functionality of OSN's designs thus raises challenges in using alternative content delivery mechanisms with higher performance.

Lockr lets OSNs use P2P systems, such as BitTorrent, to deliver content in a social networking manner. To demonstrate Lockr's versatility, we incorporated Lockr in Vuze, a popular BitTorrent client. With Lockr, Vuze supports *social torrents*: signed torrent files that specify the relationship that a downloading peer must have with the torrent's owner. Lockr's decoupling of social information and functionality makes social torrents flexible; they can be adopted easily by any online OSN to increase the speed of content delivery. Section 4.2 will describe social torrents in depth.

3. DESIGN

We begin this section by defining Lockr's key design concepts and components: personal identities, address books, social attestations, social ACLs, and techniques for revoking attestations, such as exclusion lists. We then describe how these components operate to satisfy Lockr's design goals and we demonstrate our attestation verification model in two scenarios: an online social network one and a peer-to-peer one.

3.1 Design Concepts and Components

Our design begins with the components that comprise Lockr. We describe each of these below.

3.1.1 Personal Identities and Address Books

In Lockr, a personal identity is simply a public/private keypair. Identities are generated in a decentralized manner, and individuals are in control of how these identities are shared. People can share their public keys with their social networks in the same way they share their names, addresses, and phone numbers, i.e., using business cards, e-mail signatures, letters, phone calls, or any out-of-band mechanism. They store the public keys of their friends in an address book, next to that person's other contact information. Users can refer to their friends using a locally scoped name (e.g., a nickname), which eliminates naming conflicts or name "squatting".

When issuing an attestation to a friend, a user retrieves the friend's public key from their local address book and designates it (i.e., the friend's identity) as the recipient of the attestation. Thus, people must be careful to record correctly their friends' public keys; otherwise, attestations can be issued to non-existent or even malicious identities.

3.1.2 Social Attestations

A social attestation is a piece of data that certifies a social relationship. An attestation has six fields: an issuer, a recipient, a social relationship, an expiration date, a relationship key (which we will define in the next section), and a digital signature (Figure 1 depicts an attestation in XML format). By issuing an attestation, the issuer tells a recipient that they have formed a relationship. Two parties can share more than one attestation since two people can have more than one relationship (e.g., they can be both friends and co-workers). Attestations also have an expiration date, and they are signed to prevent anyone from tampering with them.

In the common case, an attestation certifies a social relationship directly between the issuer and recipient. However,

```
<attestation>
  <issuer>Issuer's public key</issuer>
  <recipient>Recipient's public key</recipient>
  <relationship>
    <type>Relationship type (e.g., family, friend)</type>
    <firstParty>First party's public key</firstParty>
    <secondParty>Second party's public key</secondParty>
  </relationship>
  <expDate>Expiration Date</expDate>
  <relKey>Key specific to encapsulated relationship</relKey>
  <signature>Attestation's signature</signature>
</attestation>
```

Figure 1: ***The XML-based format of an attestation. An attestation has an issuer, a recipient, a social relationship between two parties, an expiration date, a relationship key, and a digital signature.***

an attestation can specify a relationship between any two parties. This is useful when a trusted third party wants to issue an attestation describing the relationship between two other entities. For example, it is perfectly reasonable for a parent to issue an attestation stating that two of their children share the "family" relationship.

3.1.3 Social Access Control Lists (ACLs)

A traditional ACL enumerates the identities of those allowed access to certain objects. In contrast, a social ACL does not rely only on identities to specify access control. Instead, they can also allow access to objects based on the *social relationship* that an individual has with the object's owner. A social ACL contains the owner's public key, the public keys of all people who can access the object (as in traditional ACLs), and a social relationship.

When a user requests access to an object protected by a social ACL, the ACL enforcer first provides the ACL to the requester. The requester then uses it to determine which attestation it should provide to the enforcer to obtain access. To access an object, the user must either: (1) have their public key listed in the social ACL, or (2) present an attestation issued to them by the owner certifying the relationship listed in the ACL. We also use XML to format social ACLs (see Figure 2).

In a social ACL, the ordering of parties specified in a relationship is important because relationships are not necessarily symmetric. Lockr requires the order in which the parties appear in an attestation to match the order in which they appear in the social ACL; otherwise, the attestation is rejected. Finally, when setting up a social ACL on a third-party Web site, people must also share the relationship key corresponding to the relationship listed in the ACL. This key lets the Web site successfully decrypt incoming attestations.

3.1.4 Techniques for Revoking Attestations

Revocation is a well-known limitation of authentication schemes based on exchanging certificates, capabilities, or attestations. In the past, much work has dealt with handling revocation in operating systems [10, 4] and distributed systems [30], and Lockr borrows techniques from this work.

We address this challenge in three ways. First, attestations let the issuer set an *expiration date*. Any system enforcing social ACLs will verify whether an attestation is still valid before granting access. Second, we augment ACLs with *exclusion lists*. An exclusion list effectively overrides valid

```
<ACL>
  <owner>Owner's public key</owner>
  <access>
    <user>User's public key</user>
                .......
    <user>User's public key</user>
   <relationship>
      <type>Relationship type (e.g., family, friend)</type>
      <firstParty>First party's public key</firstParty>
     (or <secondParty>Second party's public key</secondParty>)
    </relationship>
    (<and>, <or>, <bracket> additional relationship tags)
  </access>
</ACL>
```

Figure 2: ***The XML-based format of a social ACL. A social ACL has an object's owner, an explicit list of users who can access the content, and a social relationship that users must show to access the content. Either a "firstParty" or a "secondParty" XML attribute can be listed in the social relationship.***

attestations: it enumerates the people who cannot access the content even if they hold the appropriate attestation. Finally, users can implement revocation simply by *reissuing attestations with new relationship keys*. The downside of this approach is the need for users to deliver those new attestations to anyone who has previously been issued an attestation using that relationship. Depending on the importance of revocation, however, one might be willing to pay that price.

3.2 Design Goals for Privacy

This section describes how Lockr's components work to achieve its main design goals. These include: (1) putting users in control of their social information by decoupling it from all other functionality of an OSN site, (2) preventing OSN providers from reusing social information revealed by users when requesting access to their friends' content, and (3) improving the privacy of peers participating in a P2P online social network.

3.2.1 Decoupling Social Networking Information from OSNs

Lockr uses social attestations to encapsulate social networking information. Social attestations are OSN-independent; they can be issued and exchanged from person to person. There are many convenient ways to issue attestations, such as over e-mail or over a cell-phone Bluetooth interface. Lockr does not require the recipient to acknowledge receiving the attestation, although this could be added by a higher-level protocol.

Depending on their privacy needs, Lockr users are free to choose where to store the Lockr address books containing their social attestations. For example, users could store them on Facebook; Section 4 details our implementation of Lockr's address book as a Facebook application. Other users could opt for a more trusted Web site provider to store their address book, such as their employer. Users demanding much higher levels of privacy could even choose to run their address book on their personal computers as a standalone application. Lockr lets users decide on the level of privacy that makes them comfortable and store their address books accordingly.

Let P be the attestation's holder. P has an RSA public key (the modulus: n, and the public exponent: e), an attestation T, and an RSA signature of the attestation $T^d \bmod n$.
Let Q be the social ACL enforcer.
Step 1: P chooses 20 random positive integers (r_1, r_2, …, r_{20}). For each r_i, P computes $k_i=(r_i)^e \bmod n$ and sends them all to Q.
Step 2: Q chooses 20 random bits (b_1, b_2, …, b_{20}) and sends them all to P.
Step 3: P computes for each i, $s_i=k_i^d(T^d)^b \bmod n$ and sends them all to Q. Since each $k_i=(r_i)^e \bmod n$, then $k_i^d=((r_i)^e)^d \bmod n = r_i \bmod n$. This means that $s_i=r_i(T^d)^b \bmod n$.
Step 4: Q verifies for each i that $s_i^e=k_iT^b \bmod n$.

Figure 3: ***Our witness hiding proof of knowledge protocol of an RSA signature of an attestation. Our protocol is based on the protocol described in [21]. We used 20 integers in our implementation of Lockr, although this number can be set arbitrarily. With this setting, the probability that a malicious P can lie to Q about having the correct attestation is less than 1 in 1,000,000. We believe this trade-off is adequate in practice.***

3.2.2 Protecting Social Information

As noted, people are reluctant to expose their sensitive social relationships to third-party sites. When they have no choice, they may seek assurance that these sites will not abuse the privilege of holding this information by re-selling it to others. We designed Lockr to make social information *non-transferable*. Lockr ensures that no one can prove the validity of an attestation other than its issuer and recipient. Thus, when OSN providers learn about an attestation, they cannot transfer this information to others.

Attestations are signed with traditional digital signatures [33]. To ensure non-transferability, Lockr cannot allow a recipient to transmit a complete copy of the attestation to a third-party Web site. If it did, the Web site could store these attestations for later reuse: the social information contained in the attestation is verifiable by anyone because the attestations are digitally signed. Thus, Lockr must enable the recipient to convince a third-party Web site about the attestation's validity *without* having to transmit the attestation's signature for verification.

In cryptography, such problems are typically solved with zero-knowledge protocols [20] that provide a very strong privacy guarantee: one party can prove to another that a statement is true without revealing anything beyond the veracity of the statement. If used in Lockr, a zero-knowledge protocol for verifying the signature of a social attestation would guarantee to reveal nothing to the Web site performing the verification other than the validity of the signature. This guarantee is very strong; the Web site would never receive a copy of the signature and thus would never be able to present to others any verifiable evidence it discovers about a social relationship.

With zero-knowledge, verification is performed in rounds. In each round, the OSN site sends a challenge to the user. If he is an impostor (i.e., does not own a signed copy of the attestation), the user has a 50% chance of solving the challenge correctly. Thus, the OSN site must repeat the challenge k times to reduce the chance of cheating to 2^{-k}. This repetition makes the performance of zero-knowledge protocols impractical. Unfortunately, these protocols are not parallelizable; running all the rounds in parallel invalidates the protocols' guarantees [21].

Instead, Lockr uses a witness hiding proof of knowledge (WHPOK) protocol. WHPOK protocols are relaxations of zero-knowledge protocols. Because the WHPOK protocol we chose for Lockr is parallelizable, its performance is much more reasonable. As Section 5 will show, Lockr's WHPOK verification protocol performs reasonably well in practice: it takes only a few hundred milliseconds to complete. While WHPOK protocols meet the desired non-transferability requirement [15], they provide a slightly weaker guarantee than zero-knowledge protocols. Unlike zero-knowledge which guarantees that the Web site will not learn *any* information other than the validity of the signature, WHPOK only guarantees that the Web site is not able to learn how to prove a challenge despite verifying that the user is solving the challenge correctly. Specifically, the Web site learns about performing new computational tasks (the Step 4 in Figure 3). However, cryptographers are not able to characterize how much information is "leaked" due to WHPOK's weaker guarantee.

Lockr's attestation verification protocol is a direct implementation of the WHPOK protocol described in [21], which converts a traditional RSA signature into a WHPOK protocol. WHPOK is relatively easy to implement because it relies on traditional RSA for which there are well-known cryptographic libraries available. Another alternative to enhancing Lockr's privacy is by using anonymous credential schemes [6, 8]; however, these schemes often rely on encryption schemes other than RSA making them less easy to implement using off-the-shelf cryptographic libraries. Figure 3 presents Lockr's protocol.

3.2.3 Resolving Social Deadlock

In a decentralized scenario, two peers who do not know each other but share a common friend should be able to verify these social relationships with no loss of privacy. This scenario is challenging because each peer wants to first verify the other's social relationship before exchanging content. Lockr solves this problem through the use of relationship keys.

Lockr encrypts an attestation with its relationship key before presenting it to any other party. As its name suggests, the relationship key is specific to a particular relationship; no relationship key can belong to more than one relationship. Its purpose is to protect the confidentiality of the information contained in an attestation during the verification process. The relationship key, shared by all who have that same relationship with the issuer, must also be shared with any entity that needs to enforce a social ACL, such as a third party Web site that hosts the content. This ensures that only people who have a copy of this relationship key can decrypt the attestation.

Lockr's use of relationship keys for privacy in a decentralized scenario is inspired by *hash-based constructions* used in RE: [18, 17]. However, a simple use of relationship keys

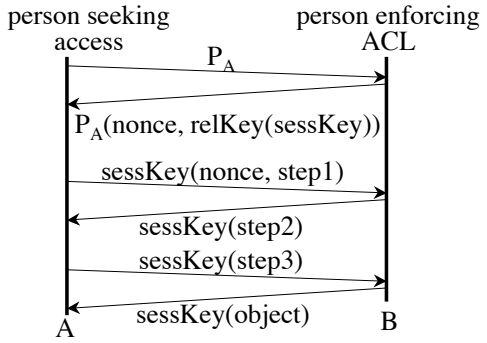


Figure 4: ***Attestation Verification for OSN sites. The arrows labeled step1, step2, and step3 correspond to WHPOK protocol steps depicted in Figure 3.***

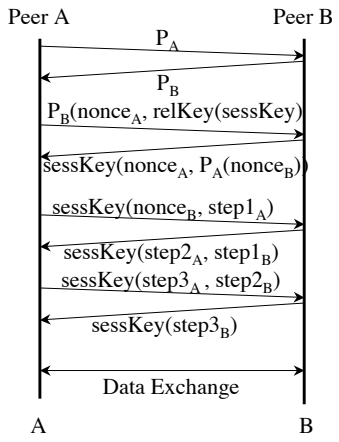


Figure 5: ***Attestation Verification for P2P. The arrows labeled step1, step2, and step3 correspond to WHPOK protocol steps depicted in Figure 3.***

leaves open a security problem in both Lockr and RE:; neither supports expiration dates. When an attestation expires, its relationship keys must become invalid. Otherwise, a peer could use expired attestations to compromise the privacy of other peers.

Lockr protects against the abuse of expired attestations by using one-way hash chains, a mechanism borrowed from TESLA, a multicast secure source authentication scheme [29]. The relationship key for a specific relationship is in fact a chain of keys, so that a different key is used on each day for all attestations containing that relationship. Each day, the recipient of an attestation must use a specific key from this chain as the attestation's relationship key for that day when presenting the attestation for validation. We assume that all participants have loosely synchronized clocks [26].

A one-way hash chain $(V_0, V_1, \ldots, V_N)$ is a collection of values such that each value V_i (except the last value V_N) is a one-way function of the next value V_{i+1}. In particular, $V_i = H(V_{i+1})$ for $0 \leq i < N$, where H is the SHA-1 hash function. When issuing an attestation, the issuer must include the relationship key valid on the day the attestation expires $V_{expDate}$. This lets the recipient obtain the relationship key for any specific day d that occurs *earlier than* the expiration date; however, the recipient cannot learn the relationship key for any day *after* the expiration date. We set N (i.e., the last date when any valid attestations can be issued) to December 31st, 2100.

3.3 Putting It All Together

Lockr's attestation verification protocols make use of the WHPOK protocol and relationship keys to offer its privacy properties. Lockr can perform two kinds of attestation verification depending on the usage scenario. In the first scenario, an OSN site verifies the attestation. In this case, the verification process is one-way: the person seeking access uses the WHPOK protocol to verify the attestation. The second is a peer-to-peer scenario. In this case, the verification process is two-way: both peers wanting to exchange content with each other use the WHPOK protocol to verify the attestation.

Attestation verification for OSN sites: This protocol has three rounds. First, the person seeking access sends his public key. The OSN site responds with a message containing a challenge (i.e., an encrypted nonce) and a session key encrypted with the relationship key. In the next round, the person answers with the resolved challenge and engages in the first step of the WHPOK protocol, as described in Figure 3. The last round corresponds to the final step of the WHPOK protocol. Figure 4 illustrates the one-way attestation verification protocol.

Attestation verification for P2P: This protocol has four rounds, and it extends the authentication scheme presented in the one-way protocol to provide two-way authentication. Figure 5 depicts the attestation verification protocol for P2P.

As noted previously, a user without an attestation can access an object as long as his identity is listed in the ACL, i.e. his public key is in the ACL. Verifying whether a user owns a private key corresponding to a public key listed in the ACL is a trivial task; for brevity, we omit describing this protocol.

3.4 Security Limitations

We anticipate two main issues with Lockr's security model. First, the recipient of an attestation can choose to reveal the relationship key to others or even collude with an attacker. In such a case, Lockr's privacy properties are compromised. However, we believe that "social pressure" will act as a deterrent to these types of privacy attacks because the recipient of an attestation *does* have a social relationship with the issuer. These types of attacks are analogous to the ones in which somebody's friend "breaks the silence" by making public the identities of other friends.

Another possibility is that an attestation's recipient decides to sell the attestation for monetary reward. When doing so, the seller must also share the private key along with the attestation. However, sharing the private key compromises *all* other attestations held by the recipient. This raises the cost incurred by the recipient for selling their attestation – this person relinquishes control over *all* their attestations, including the ones they will receive in the future using the same private key as their identity.

4. IMPLEMENTATION

We implemented Lockr for the BitTorrent P2P file sharing system and the Flickr photo sharing site; our implementation for Flickr requires neither Flickr's cooperation

LockrCenter Facebook API
getKeyPair(): returns user's OpenSSL 1024-bit RSA key-pair
getAttestations(publicKey): returns user's attestations issued by publicKey
getAllAttestations(): returns all user's attestations
putAttestation(): stores an attestation
verifyAttestation(attestation, acl): verifies whether the attestation is authentic and it satisfies the given ACL rules

Table 1: ***The Lockr Center API for storing and retrieving attestations. Lockr Center uses the Facebook's DataStore to store a user's private key, public key, and the set of all received attestations in the user's Facebook account.***

nor server-side support. Before describing these implementations, however, we discuss Lockr Center. Plug-ins for both applications contact the Lockr Center address book to retrieve and store social attestations. Our implementation code and executables are available for download from `www.lockr.org`.

4.1 Lockr Center: An Address Book for Lockr

People use address books to store the identities of those in their social network. Lockr's address book has two additional roles: (1) to allow people to exchange attestations with each other, and (2) to allow any online application to store and retrieve attestations on behalf of its users. Any existing address book, such as Gmail's My Contacts or iPhone's Contacts, can be easily enhanced to support this functionality. Users can choose their favorite address book for use with Lockr; the choice is unimportant as long as the address book conforms to the format of the attestations and the protocols for accessing them. Users can even create and manage their own address books without relying on third party providers to provide them.

We implemented Lockr Center, an address book for Lockr, as a Facebook application. Lockr Center stores a user's private key, public key, and all attestations the user has received from others using the Facebook Data Store [14]. Any application can use Facebook's API to retrieve any stored attestations after the user has logged in to Facebook. Table 1 presents Lockr Center's Facebook API for storing and retrieving attestations.

As with any other Facebook application, users must first add Lockr Center to their set of enabled Facebook applications in order to issue and receive attestations. They can then issue attestations to any other Facebook user running Lockr Center. Users can also invite others to install Lockr Center by issuing Facebook notifications (only to other Facebook users) or by sending e-mail invites. Figure 6 illustrates the user interface to Lockr Center.

4.2 Lockr for BitTorrent

We wrote a Lockr plug-in for Vuze (v 3.0.3.5)[1], the popular BitTorrent client application, and tested it on both Windows and Linux. The plug-in modifies Vuze by extending the file format for torrent files to include support for social ACLs and by adding an attestation exchange and verification step during the formation of BitTorrent connections between peers.

[1]Vuze was formerly known as Azureus.

Figure 6: ***The Lockr Center address book.***

4.2.1 A BitTorrent Primer

A *torrent file* is a collection of key-value pairs describing the information needed to begin downloading a file from other BitTorrent peers. This information includes the file's meta-data as well as the IP addresses and port numbers of one or more BitTorrent trackers. A *tracker* is a server that keeps track of the active peers participating in a BitTorrent download. By contacting a torrent's tracker, a peer can find other peers serving parts of the respective file. BitTorrent trackers are not trusted because anyone can decide to become a tracker, and BitTorrent provides no mechanisms to protect against malicious trackers.

To download a file, a BitTorrent peer contacts a tracker from the torrent file. The tracker returns a list of available peers that can serve fragments of the content. The downloader then initiates TCP connections to these peers. Once a TCP connection is established, the peers engage in an application-level handshake to determine which fragments each can serve the other. If both peers run the Vuze BitTorrent client, this initial handshake is followed by a second handshake to determine which Vuze options each peer supports, such as encryption or chat messaging functionality.

4.2.2 Social Torrents

A *social torrent* is a torrent file with two new key-value pairs: a social ACL and a digital signature. The social ACL has the XML format illustrated in Figure 2. The file's owner signs the torrent file to prevent a tracker from altering it (although the signature does not cover the list of trackers so that it can be modified). We modified Vuze's torrent creation module to support the creation of social torrents.

Because BitTorrent trackers cannot be trusted, Lockr instead relies on individual peers to enforce social ACLs. Our design choice works well for sharing content in social networks because peers are subject to "social pressure" that discourages them from becoming malicious. This illustrates another Lockr benefit: dealing with trust in social networks is much simpler than it is in global, Internet-wide settings.

4.2.3 Social Handshake

We implemented Lockr functionality as a plug-in to Vuze that implements an extension to the BitTorrent handshake protocol. If both peers implement Lockr, they engage in a social handshake using the WHPOK protocol in addition to the BitTorrent handshake performed on connection setup.

Social Vuze Messages
LockrPubKey(): sends a peer's public key
LockrOutgoingChallenge (): sends a nonce and a session key; the session key is encrypted with the relationship key; everything is encrypted with the other peer's public key
LockrIncomingChallengeOutgoingResponse(): sends another nonce and the received nonce from the other peer; the local nonce is encrypted with the other peer's public key; everything is encrypted with the session key
LockrWHPOKOutgoingCommitIncomingResponse (): sends the 20 positive integers of the 1st step of WHPOK and the received nonce from the other peer; everything is encrypted with the session key
LockrWHPOKIncomingCommitOutgoingChallenge(): sends 20 positive integers of the 1st step of WHPOK of the local peer and 20 bits of the 2nd step of WHPOK of the remote peer; everything is encrypted with the session key
LockrWHPOKOutgoingResponseIncomingChallenge(): sends the 20 integers computed in the 3rd step WHPOK of the local peer and 20 bits of the 2nd WHPOK of the remote peer; everything is encrypted with the session key
LockrWHPOKIncomingResponse(): sends the 20 integers computed in the 3rd step of WHPOK of the local peer encrypted with the session key

Table 2: ***Seven extra messages supported by Lockr for Vuze. A peer uses these messages to extend the Vuze handshake with a social handshake. These messages have a 1:1 correspondence with the attestation verification protocol shown in Figure 5.***

This social handshake implements the two-way attestation verification protocol using seven message types we added to Vuze, shown in Table 2.

Once the social handshake completes successfully, peers start the file transfer. File contents are encrypted using the session key exchanged in the social handshake. If the social handshake does not complete successfully (e.g., the ACL verification fails or one of the peers does not support Lockr), the connection is immediately terminated. Note that Lockr-enabled peers engage in a social handshake for social torrents only. For files with regular torrents, Lockr-enabled peers do not exchange Lockr-specific messages. In this way, Lockr for Vuze clients remain backward-compatible with other Vuze clients.

4.3 Lockr for Flickr

Implementing Lockr for Flickr would normally require Flickr's cooperation to implement the one-way attestation verification protocol. Absent server-side support, we decided to implement a working prototype by delegating the social ACL enforcement to a proxy server hosted at the University of Toronto. We made this inelegant design decision to illustrate Lockr's benefits when sharing photos on Flickr. Our workaround still decouples the social network from the content sharing site – people's photos are still stored on Flickr's servers.

On the client side, we implemented a Firefox plug-in written in JavaScript for two reasons. First, it extends the interface for uploading pictures to flickr.com to let users create social ACLs. Second, the plug-in communicates with the proxy server to perform attestation verification whenever the user visits Flickr photos that are protected by Lockr.

4.3.1 Implementation Details

To protect a photo with Lockr, the plug-in uploads a dummy photo to flickr.com in addition to the original photo. The plug-in sets the permissions on the original photo to "private". In Flickr, this setting makes the photo accessible only through a URL that is hard to guess. Our plug-in sends a copy of this hard-to-guess URL to our proxy, and the proxy can return this URL to anyone who presents a valid attestation. The dummy photo is set for public viewing. Without our plug-in or appropriate Lockr attestations, visitors to flickr.com can view the dummy photo only.

Upon starting the browser, our plug-in contacts Lockr Center to download and cache all the user's social attestations. Next, the plug-in authenticates to our proxy server but sends no attestations. These operations persist across the browsing session. We chose to implement these operations during startup rather than upon browsing flickr.com because we believe users are less tolerant of delays during the latter.

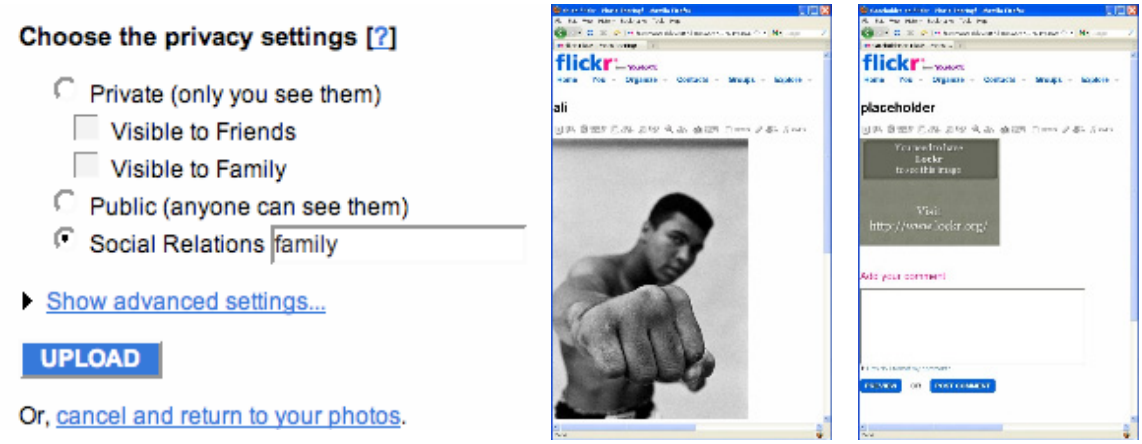
When visiting a page on flickr.com that stores the dummy photo, our plug-in sends the required attestation to our proxy. We do not use the WHPOK attestation verification protocols when implementing Lockr for Flickr (a choice we explain in the next Section). The proxy verifies the attestation and returns the hard-to-guess URL that indicates where the original photo is located. The plug-in automatically swaps the dummy and original photos in a manner completely transparent to the user.

Figure 7 illustrates the plug-in's user interface. Figure 7a shows an ACL selection done with Lockr when a picture is being uploaded to Flickr. Users can make the content "private", "public", or protected with "social relationships". The first two options are part of Flickr's default behavior; our plug-in adds the last option. Figure 7b shows the original image when viewed by a user with the required attestations. Figure 7c shows a dummy image seen by a user without the required attestations (or without an installed plug-in).

4.3.2 Implications of Our Implementation

Our implementation for Flickr deviates from Lockr's design by not using the WHPOK protocols. Instead, it sends attestations directly to our proxy. We made this implementation decision due to a limitation of the JavaScript environment: the current cryptographic support in JavaScript is incomplete and slow. One practical way to make our plug-in perform the cryptography locally is by adding platform-specific cryptographic libraries to it; however, we believe that doing so will deter the adoption of our plug-in. Instead, the plug-in delegates this computation to our proxy.

Despite the lack of server-side support, our integration of Lockr with Flickr is secure. Photos protected by social ACLs cannot be viewed by users without our plug-in or without the required attestations. The security of our plug-in rests on Flickr's implementation of hard-to-guess URLs for private content. However, not all Web 2.0 sites offer this op-



(a) (b) (c)

Figure 7: ***Lockr for Flickr. (a) A user creates a social ACL. (b) A user views the protected image with the appropriate attestation. (c) A user without appropriate attestations sees a dummy image.***

tion. Another common way to make content private is by asking users to login before requesting access (e.g., Google Picasa uses this mechanism). Once the user logs on, the Web site can decide whether the user-id is allowed to access the private content.

Even with this alternate mechanism for protecting private content, we believe it is possible to implement Lockr in the absence of server-side support. To do so, our proxy needs its own user-id on the respective Web site. Lockr users would have to designate this account's user-id as one that has access to their private content. In this way, our proxy can still mediate access to private content and fetch it on behalf of Lockr users with the required attestations.

5. EVALUATION

This section evaluates Lockr's performance and overhead with respect to two different and popular sharing systems, BitTorrent and Flickr. We can effectively evaluate Lockr's usability only after it has been deployed more widely in the field. Instead, we evaluate the performance of both our implementations and we demonstrate that Lockr's overhead is negligible in practice.

In all our experiments, we ran Lockr on a laptop equipped with an Intel Pentium M CPU and 1.5GB of RAM, a typical configuration for today's users. We instrumented all code to record the time spent in different parts of Lockr's implementation. All experiments were repeated 20 times, and we report the average of these 20 measurements. The remainder of this section evaluates how long it took for Lockr to issue an attestation as well as Lockr's performance when integrated with Flickr and BitTorrent.

5.1 Issuing an Attestation

Issuing an attestation requires two operations: (1) computing the relationship key using a one-way hash chain, and (2) signing the attestation. The relationship key is computed by repeatedly hashing the attestation's relationship with SHA-1; the number of hashing operations equals the number of days between the attestation's expiration date and the last date of a hash chain (currently set to December 31st, 2100). To measure worst-case performance, we issued an attestation that was valid for one day only; for this attestation, we performed 33,840 repeated hashes to compute the relationship key. Even for this worst-case, we found that Lockr users will observe no noticeable delay when issuing attestations: generating the relationship key took 54.48ms, and signing the whole attestation took 3.6ms, for a total of 58.08ms.

5.2 Lockr for Flickr

We implemented Lockr for Flickr by delegating ACL enforcement to one of our own servers hosted at the University of Toronto. Upon starting the browser, the Lockr plug-in authenticated to our server and retrieved all the user's attestations, caching them on the local machine for later use. These operations were done once only, at the beginning of each browsing session. Our experiments found this one-time overhead to be relatively small. Authenticating to our server using the protocol described in Figure 4 took 471.8ms on average, and retrieving the attestations took an additional 778.3ms, for a total time of about 1.25 seconds.

Each time a user visits a flickr.com page that stores content protected by Lockr, the plug-in fetches the social ACL and finds the corresponding attestation in the local cache. The plug-in sends this attestation back to our server (Lockr for Flickr does not use the WHPOK protocols) and receives a hard-to-guess URL on flickr.com where the content is stored. We found that the overhead of this entire chain of operations took 104.4ms.

5.3 Lockr for BitTorrent

As described in Section 4.2.3, the implementation of the social handshake in Lockr for BitTorrent uses seven new message types. The social handshake is initiated by peer A to peer B. Table 3 breaks down its performance overhead. Note that these numbers do not include the latency for delivering messages because latency depends on the distance between nodes; instead, we show the computational overhead at each node. The social handshake added at most a latency equivalent of four RTTs to the BitTorrent handshake. As Table 3 shows, social handshake overhead was small: Lockr added 880.25ms to the setup time of a BitTorrent connection between peers.

	Speed (ms)
Peer A: LockrPubKey	24.07
Peer B: LockrPubKey	24.07
Peer A: LockrOutgoingChallenge	583.70
Peer B: LockrIncomingChallengeOutgoingResponse	60.47
Peer A: LockrWHPOKOutgoingCommitIncomingResponse	51.59
Peer B: LockrWHPOKIncomingCommitOutgoingChallenge	55.51
Peer A: LockrWHPOKOutgoingResponseIncomingChallenge	39.44
Peer B: LockrWHPOKIncomingResponse	41.40

Table 3: ***Speed (ms) of the operations performed in a social handshake in Lockr for BitTorrent. These numbers include only local processing times, not network latencies. Overall, Lockr's social handshake took 880.25ms to complete.***

6. RELATED WORK

Lockr's concepts are related to previous privacy protocols and authentication systems. This section presents work on protocols with strong privacy properties and work that uses social networks to bootstrap trust in online systems. It also relates Lockr to previous authentication and access control systems.

6.1 Privacy Properties

Lockr's WHPOK attestation verification is a relaxation of the more general class of zero-knowledge protocols [20]. While zero-knowledge protocols offer very strong privacy properties, their expense makes them impractical. Although WHPOK protocols offer less privacy protection than zero-knowledge protocols, they provide adequate performance while protecting the confidentiality and non-transferability of information. For example, the Web site enforcing a social ACL with Lockr can prove that *someone* presented a valid social attestation. This differs from non-transferability; it is impossible for the Web site to identify precisely who presented it. For a rigorous description of the differences between zero-knowledge and WHPOK, please see [15].

6.2 Using Social Networks to Bootstrap Trust

Reliable Email (RE:) [18] is a white-listing system for email that incurs zero false positives among socially connected users. It exploits Friend-of-Friend (FoF) relationships among correspondents to populate white-lists. RE: suffers from several privacy issues pointed out in [17]. In particular, like Lockr, RE: users must discover FoF relationships even when they do not trust each other. The *hash-based construction* proposed to address this problem [17] differs from Lockr's in three ways. First, the trust model differs. [17] assumes that both parties exchange a secret when an attestation is issued; in contrast, Lockr lets one person issue an attestation without requiring a secret in return. Second, although attestations in [17] include expiration dates, the protocols do not support them, and adding such support is not trivial. Finally, [17] requires parties to exchange signed attestations, whereas Lockr uses WHPOK protocols.

Authenticatr [31] is a system for bootstrapping authenticated communication channels using social networks. Authenticatr's goal is very different from Lockr's: it seeks to incorporate social information from today's online social networks into any online application. In this way, online applications can use online social networks to bootstrap trust. Authenticatr's design differs from Lockr's, as well. It provides a middleware layer that can query several online social networks and expose this information to any application through a social networking API.

NOYB [22] and **Persona** [2] are online social networks that share one of Lockr's goals: they put users in control of their own social information. NOYB obfuscates a user's sensitive data by using a secret function to permute the data of all OSN users. This permutation effectively "mixes" different information from different profiles randomly. Each user possesses a part of the secret function that allows them to reconstruct their private information. A user can effectively share their secret with their friends allowing them to view the user's private information. On the other hand, Persona uses a cryptographic mechanism to achieve the same goal; its attribute-based encryption (ABE) lets users apply fine-grained access control policies to their data. With ABE, users encrypt their data with encryption keys that are relationship-specific. Persona does not provide support for expiring relationship keys. Instead, to revoke a relationship key, such as "friend", Persona requires a fresh re-keying of all remaining friends. In Persona and NOYB, OSNs store only encrypted content. This makes it more difficult for them to provide functionality, which requires access to raw content. Unlike Lockr, Persona and NOYB do not guarantee the non-transferability of the social information disclosed during access requests.

6.3 Authentication and Access Control

Capabilities [12, 25] are secure tokens that enable specific access rights to an object or resource. Recent projects have proposed using capabilities to manage and share personal data online [19, 23]. There are two important differences between capabilities and Lockr's social attestations. First, capabilities encapsulate a unique object identifier and access rights; thus, the tasks that can be performed by holding a capability are known in advance. In contrast, social attestations are application-independent: they are issued without any prior knowledge about how they will be used. This property makes Lockr more flexible. People can specify different access control policies on different systems without the need to issue new attestations.

Second, capabilities are inherently transferable, while attestations are restricted to the recipient only. This difference has implications with respect to delegation, viz., that capability-based systems naturally support delegation, while attestations do not.

SPKI/SDSI [13, 32] is a certificate-based PKI (like X.509) with a decentralized design. In the absence of a trusted third party, SPKI/SDSI users must provide a chain of certificates to a trusted party or group to authenticate themselves. While Lockr uses one-hop social networking authentication, SPKI/SDSI implementations often have complicated chain discovery mechanisms [9] that raise deployment difficulties. Also, some of the distinctions between Lockr and X.509 apply to SPKI/SDSI, e.g., finding a secure channel and the lack of support for privacy.

Role-based access control [16, 5] is an approach to re-

stricting access to resources based on roles. Users are assigned various roles, and these roles dictate what permissions users acquire. Lockr's access control can be viewed as a form of role-based access control because one can view a social relationship as a way to express a role. However, Lockr is designed to address the needs of social networks, whereas role-based access control was developed for use inside organizations [16]. Further, role-based access control systems have global namespaces and role hierarchies [5], unlike Lockr.

PGP [34] is another decentralized authentication system that uses a vetting scheme in which users sign each other's public keys. To verify a signature, a user must find a chain of trust linking the owner to themselves. Over time, PGP creates a "Web of trust" in which people accumulate each other's verified signatures.

Lockr differs from PGP in two ways. PGP's notion of trust is all or nothing, whereas Lockr uses social relationships that lets people express different nuances of trust. Second, PGP's trust is transitive. However, since social relationships are diverse and complex in real-life, Lockr does not combine them to form transitive relationships.

OpenID [28] is a decentralized single sign-on system. Using OpenID-enabled Web sites, a user need not remember multiple usernames for different Web sites. Instead, he can register with an OpenID "identity provider", and anyone can be such a provider. The OpenID is a globally unique identifier tied to the user's chosen provider.

There are two important differences between Lockr and OpenID. First, OpenID's namespace is global, and names are globally unique; in contrast, Lockr's attestations are social networking-scoped. Second, OpenID is vulnerable to phishing attacks [36]. An attacker can attempt to phish an OpenID provider; if successful, the attacker can use the stolen identity on any OpenID-enabled Web site. Lockr is not vulnerable to phishing because of its challenge-response mechanism. Its users must use their private keys to resolve a challenge every time they request access.

7. CONCLUSIONS

This paper presented Lockr, an access control system using social networking abstractions. Lockr was designed to simplify content sharing on the Internet. It decouples the management of social information from online sharing systems by letting users exchange application-independent attestations. This decoupling facilitates the integration of Lockr's access control with any online application, such as Web 2.0 sites and P2P file sharing.

This paper described the design and implementation of Lockr in the context of two online systems with very different characteristics: a centralized Web 2.0 site, Flickr, and a decentralized P2P system, BitTorrent. Our access control uses verification protocols with attractive privacy properties that reduce the chance of third-party sites abusing people's personal information, such as their social relationships.

Our implementation demonstrated that Lockr can be deployed even in the absence of cooperation from Web sites. This permits experimentation with Lockr's access control while we continue to share content using existing sites. Our initial experience using Lockr on Flickr and on BitTorrent has confirmed that Lockr simplifies management and access control for content sharing.

Acknowledgments

We thank Charlie Rackoff for initiating us in the subtleties of zero-knowledge protocols. Some of the initial ideas in Lockr originated during discussions with Kiran Gollu. Geoffrey Salmon contributed to Lockr's codebase. Finally, we thank Krishna Gummadi and the anonymous reviewers for their helpful comments.

8. REFERENCES

[1] M. Aspan. How sticky is membership on facebook? just try breaking free, 11 February 2008. New York Times.

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

[3] BBC News. Brown apologises for records loss, November 2007. `http://news.bbc.co.uk/1/hi/uk_politics/7104945.stm`.

[4] V. Berstis. Security and protection of data in the ibm system/38. In *Proc. of the 7th Annual Symposium on Computer Architecture (ISCA)*, La Baule, France, May 1980.

[5] A. Bhora, S. Smaldone, and L. Iftode. FRAC: Implementing role-based access control for network file systems. In *Proc. of the 6th IEEE Symposium on Network Computing and Applications (NCA)*, Baltimore, MD, July 2007.

[6] S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. The MIT Press, 2000.

[7] S. Buchegger and A. Datta. A case for p2p infrastructure for social networks - opportunities and challenges. In *In 6th International Conference on Wireless On-demand Network Systems and Services (WONS)*, Snowbird, UT, Feb. 2009.

[8] J. Camenisch and A. Lysyanskaya. Signature schemes with efficient protocols. In *3rd International Conference on Security in Communication Networks (SCN)*, Amalfi, Italy, September 2002.

[9] D. Clarke. SPKI/SDSI HTTP server/certificate chain discovery in SPKI/SDSI, September 2001. Masters thesis, Massachusetts Institute of Technology.

[10] E. Cohen and D. Jefferson. Protection in the hydra operating system. In *Proc. of the 5th Symp. on Operating Systems Principles (SOSP)*, Austin, TX, November 1975.

[11] L. A. Cutillo, R. Molva, and T. Strufe. Privacy preserving social networking through decentralization. In *In 6th International Conference on Wireless On-demand Network Systems and Services (WONS)*, Snowbird, UT, Feb. 2009.

[12] J. B. Dennis and E. C. V. Horn. Programming semantics for multiprogrammed computations. *Communications of the ACM*, 9:143–155, March 1966.

[13] C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. M. Thomas, and T. Ylonen. Spki certificate documentation, 2001. `http://world.std.com/~cme/html/spki.html`.

[14] Facebook Developers WiKi. Data Store API Documentation, 2008. http://wiki.developers.facebook.com/index.php/Data_Store_API_documentation.

[15] U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, Baltimore, MD, May 1990.

[16] D. F. Ferraiolo and D. R. Kuhn. Role-based access controls. In *Proceedings of the 15th National Security Conference*, Baltimore, MD, October 1992.

[17] M. J. Freedman and A. Nicolosi. Efficient private techniques for verifying social proximity. In *Proc. of 6th Workshop on P2P Systems*, Bellevue, WA, Feb 2007.

[18] S. Garriss, M. Kaminsky, M. J. Freedman, B. Karp, D. Mazieres, and H. Yu. Re: Reliable email. In *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, May 2006.

[19] R. Geambasu, M. Balazinska, S. D. Gribble, and H. M. Levy. Homeviews: Peer-to-peer middleware for personal data sharing applications. In *Proc. of SIGMOD Conference on Management of Data*, Beijing, China, June 2007.

[20] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the 17th Symposium on the Theory of Computation*, Providence, Rhode Island, May 1985.

[21] S. Goldwasser and E. Waisbard. Efficient transformation of well known signature schemes into designated confirmer signature schemes. *Technical Report MCS03-13, The Weizmann Institute of Science*, 2003.

[22] S. Guha, K. Tang, and P. Francis. NOYB: Privacy in Online Social Networks. In *Proceedings of the 1st ACM Sigcomm Workshop on Online Social Networks (WOSN)*, Seattle, WA, USA, August 2008.

[23] P. J. Keleher, N. Spring, and B. Bhattacharjee. Chit-based access control. Technical Report CS-TR-4878, University of Maryland at College Park, 2007.

[24] B. Krishnamurthy and C. Wills. On the leakage of personally identifiable information via online social networks. In *Proceedings of the 2nd ACM Sigcomm Workshop on Online Social Networks (WOSN)*, Barcelona, Spain, August 2009.

[25] H. M. Levy. *Capability-Based Computer Systems*. Butterworth-Heinemann, Newton, MA, USA, 1984.

[26] D. L. Mills. RFC 1305: Network Time Protocol (Version 3) Specification, Implementation and Analysis, 1992. http://tools.ietf.org/html/rfc1305.

[27] A. Noyes. Facebook averts ftc privacy complaint, 23 February 2009. Tech Daily Dose.

[28] OpenID. OpenID, 2008. http://openid.net/.

[29] A. Perrig, R. Canetti, D. Song, and J. D. Tygar. Efficient and secure source authentication for multicast. In *Network and Distributed System Security Symposium (NDSS)*, pages 35–46, Feb. 2001.

[30] B. C. Popescu, B. Crispo, and A. S. Tanenbaum. A certificate revocation scheme for a large-scale highly replicated distributed system. In *Proc. of the 8th Symp. on Computers and Communications (ISCC)*, Kemer, Turkey, July 2003.

[31] A. Ramachandran and N. Feamster. Authenticated out-of-band communication over social links. In *Proc. of the 1st ACM SIGCOMM Workshop on Online Social Networks (WOSN)*, Seattle, WA, August 2008.

[32] R. L. Rivest and B. Lampson. SDSI 2.0 - a simple distributed security infrastructure, 1997. http://groups.csail.mit.edu/cis/sdsi.html.

[33] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[34] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. Wiley; 2nd edition, 1995.

[35] A. Shakimov, A. Varshavsky, L. Cox, and R. Caceres. Privacy, cost, and availability tradeoffs in decentralized osns. In *Proceedings of the Workshop on Online Social Networks (WOSN)*, Barcelona, Spain, Aug 2009.

[36] M. Slot. Beginner's guide to OpenID phishing. http://openid.marcoslot.net/.

[37] B. Stone and B. Stelter. Facebook withdraws changes in data use, 18 February 2009. New York Times.

[38] T. Ullyot. Results of the inaugural facebook site governance vote, 23 April 2009. Facebook.

[39] J. Vascellaro. Facebook's about-face on data, 19 February 2009. Wall Street Journal.