

# Lockr: Social Access Control for Web 2.0

Amin Tootoonchian<sup>†</sup>, Kiran K. Gollu<sup>†</sup>, Stefan Saroiu<sup>†</sup>, Yashar Ganjali<sup>†</sup>, Alec Wolman<sup>‡</sup>

<sup>†</sup>Department of Computer Science  
University of Toronto  
Toronto, ON M5S 2E4

{amin,kkgollu,stefan,yganjali}@cs.toronto.edu

<sup>‡</sup>Microsoft Research  
Redmond, WA 98052

alecw@microsoft.com

## ABSTRACT

Sharing personal content online is surprisingly hard despite the recent emergence of a huge number of content sharing systems and sites. These systems suffer from several drawbacks: they each have a different way of providing access control which cannot be used with other systems; moving to a new system is a lengthy process and requires registration and invitation of all one's friends to the new system; and the rules for access control are complicated and become more so as our networks of online friends grow.

In this paper, we present Lockr – an access control scheme based on social relationships that makes sharing personal content easy. Lockr separates social networking information from the content sharing mechanisms, thereby eliminating the need for users to maintain many site-specific copies of their social networks. We describe Lockr's design, security properties, and limitations. We also present how we integrated Lockr with two popular systems for sharing content online – BitTorrent and Flickr.

## Categories and Subject Descriptors

C.2.4 [Computer-Communication Networks]: Distributed Systems

## General Terms

Design, Security

## Keywords

BitTorrent, Lockr, Web, access control, online social networks

## 1. INTRODUCTION

Today, sharing personal content is surprisingly difficult. Current systems suffer from a number of drawbacks. They are cumbersome to use, they impose artificial limits on the size of shared content (e.g., pictures and video), and they make it difficult to restrict content only to a specific set of users. For those Web sites that do provide access control, they typically require all the participants to be registered with the site in question. This imposes the burden that

users must register with many sites, and maintain separate and potentially inconsistent copies of their social networks for each site.

Despite all these problems, content sharing Web sites and services remain very popular. These content sharing sites account for half of the top 10 most visited Web sites today [1]. Together these sites attract millions of visitors every year serving petabytes of personal content. These sites' popularity have amplified their shortcomings – today's Web users need to register, create, and maintain dozens of copies of social networks on dozens of sites just to share their personal content.

This paper's contributions are two-fold: we present the design of Lockr – an access control scheme that makes sharing personal content easy and we describe how we integrated Lockr with BitTorrent and Flickr, two popular systems for sharing content online. Lockr is based on a simple insight – we must decouple content delivery and sharing from managing social networking information. Our scheme lets people manage their social networks themselves in one place (e.g., through their personal address books) while letting Web sites and Internet systems be in charge of content delivery only. For this, we introduce two new concepts – social attestations and social access control lists (ACLs). At a high-level, a social attestation is a small piece of meta-data issued by one person to another encapsulating a social relationship. The recipient can use this attestation to prove the social relationship to any online site or to any other user. People exchange these attestations *once* while reusing them to gain access to their friends' personal content scattered across different sites and systems. Unlike a capability, an attestation does not include access rights. Instead, access rights are specified in the social ACL by describing what social relationships a person must have to gain access to the data.

To illustrate how Lockr works, consider a simple example of a person wanting to restrict access to their family photos on Flickr. The owner creates a social ACL indicating that access to the photos is restricted to family-only. Family members must present their social attestations to Flickr issued by the photos' owner before gaining access. To allow access, Flickr must verify that the attestations were issued by the original owner and that "family member" is the social relationship encapsulated by the attestation. Note that the family members' social attestations can be reused by any online site without requiring users to register.

Lockr allows users to express access control policies based on social relationships. This eliminates the need to manage many site-specific social networks online. Users need to manage a single social network that can be stored in an address book on their own machines. To create an access control policy, users do not have to enumerate all members of a social group. Instead, they can list what social relationships others must have to gain access to personal content. We exclude access rights, application semantics,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WOSN'08, August 18, 2008, Seattle, Washington, USA.

Copyright 2008 ACM 978-1-60558-182-8/08/08 ...\$5.00.

and object names from attestations unlike other recently proposed capability-based schemes [4, 5]. In this way, attestations can be re-used across different Web sites and Internet systems. This usage model conforms with the familiar model with which users already manage contacts in their address books making Lockr intuitive and easy-to-use.

Lockr is incrementally deployable – any online site can support it without coordinating with other sites. The real benefit from Lockr, however, is gained when a large number of sites deploy and support it, as users will not need to define and manage their networks for such sites. As our second contribution in this paper, we show how one can implement Lockr on a content sharing site without any support from that site. As an example, we show how one can use Lockr on the popular photo sharing site Flickr, using a browser plug-in. Clearly, this is not the long-term solution we envision, but is a simple way to show the advantages of Lockr. We also present an implementation of Lockr in Azureus, a popular BitTorrent client, providing access control for the content shared in a peer-to-peer (P2P) system.

## 2. REQUIREMENTS FOR ACCESS CONTROL ON PERSONAL CONTENT

Because personal content is usually of a private nature, people need to restrict who can access it. While some delivery systems today provide access control, these schemes do not fit the needs of personal content. We believe that an access control scheme for sharing personal content online should have the following properties.

**1. Provide a simple and flexible way of defining access control rules.** The growth of our online social networks will make it more and more difficult to manage who has access to which online content. An ideal access control scheme must be intuitive and simple to use, matching the way in which people manage their social networks both online and offline.

**2. Eliminate the management issues associated with using many content sharing sites.** The volume of personal content created and shared online has immensely grown, and will continue to grow. Depending on the type of content, users might prefer using different types of services. An ideal access control scheme must be able to work with all types of content regardless of where they are stored. Users should not have to manage many copies of their social networks and they should not have to convince their friends and families to register on many different sites.

Our proposed access control scheme captures these properties using two simple observations.

**1. We must use social relationships to describe access control policies.** Our observation is that it is very intuitive for people to use their social relationships to define access control policies for their personal content. For example, people want to share family pictures with family members only, they want to share pictures from work with work colleagues, and they want to share their wedding videos with all the guests to their wedding. In contrast, today's sharing systems provide access control mechanisms based on identities and groups that vary from one content sharing service to another. While access control policies based on identities and groups are adequate for operating systems or databases, we believe they are very difficult to manage over different online content sharing systems. People are more comfortable restricting access to their pictures to their "family" or "work colleagues". It is more intuitive, and one doesn't need to enumerate people's identities for every piece of data to pro-

tect. With Lockr, people can use social relationships (in addition to identities) to define access control policies for personal content.

**2. We must separate social networks from content delivery and sharing.** A social access control scheme should be compatible with any Internet system for delivering personal content. Internet systems must decouple their social networking information from their content delivery and sharing functionality. With Lockr, people manage one single copy of their social network stored either on their own desktops or on a third-party site, whichever is more convenient.

Separating social networks from content delivery and sharing opens a new realm of possibilities for constructing applications that have yet to emerge on the Internet. For example, today's firewall policies filter incoming traffic based on IP addresses, port numbers, or protocol types. Instead, firewalls could implement filtering policies based on social relationships. A personal firewall could allow incoming traffic from others based on their social relationship to the firewall's owner and not based on their IP addresses. Similarly, organizations could implement policies allowing traffic from their employees only and without relying on IP addresses. As another example, a person's e-mail reader can share locally archived e-mails with this person's family or friends. The e-mail reader could also implement a spam classification scheme based on the social relationships between senders and receivers, similar to a system like RE: [3].

## 3. DESIGN

To provide the requirements described previously, Lockr introduces two new concepts – social attestations and social access control lists (ACLs). In this section, we start by describing how people create a personal identity in Lockr. We then present the concept of social attestations, and how they are issued and exchanged. We describe social ACLs and how they are enforced. Finally, we discuss the issues raised by revoking these attestations.

### 3.1 Personal Identities and Address Books

In Lockr, a personal identity is a pair of a public key and a private key. People communicate their public keys with their social networks in the same way they share their names, addresses and phone numbers – using business cards, e-mail signatures, letters, phone calls, or any out-of-band mechanisms. Public keys are stored in address books, next to a person's other contact information. While public keys can be revealed to anyone, their exchange must be done securely. An adversary can impersonate a victim once the victim's friends record an incorrect public key in their address books. However, these keys are exchanged only by people forming social relationships and not between unknown strangers. This allows people to find convenient ways to exchange their public keys securely, such as secure e-mail or over their cell-phones' Bluetooth interfaces.

### 3.2 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 between two parties, an expiration date, a relationship key, and a digital signature. The meaning of an attestation is that an issuer tells a recipient that two parties form a relationship. Two parties could share more than one attestation since two people can have more than one relationship (e.g., two people can be both friends and co-workers). In the simplest case (and we believe the most common), an attestation certifies a social relationship between the issuer and the recipient. For example, when a person issues

```

<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. The identities of the issuer, the recipient, and the two parties are represented with public keys. All attestations issued by the same issuer and encapsulating the same relationship must have the same relationship key.*

social attestations to a family member, the attestation certifies the relationship "family" between the issuer and the recipient. While it is common for the issuer and the recipient to be the two parties forming the relationship, the relationship can be between any two parties in the most general case. Attestations also have an expiration date or they can be set to never expire. Finally, attestations are signed to prevent anyone from tampering with them.

The relationship key is a shared key among all parties with the same relationship with the attestation's issuer. Its role is to protect attestations from being revealed to third parties. Whenever the attestation is transmitted to another party, it is first encrypted with the relationship key. This prevents malicious third parties from having access the social information encapsulated in attestations even if they were to intercept their transfers illegitimately. We will describe how relationship keys are used in Section 3.3.1.

There are many convenient ways in which attestations can be transmitted to a recipient, such as e-mail or over cell-phones' Bluetooth interfaces. Attestations can also be exchanged over a variety of different application protocols, such as HTTP or chat protocols. Since the issuer and the recipient have each others' public keys in their address books, the attestation is encrypted with the recipient's public key before being sent. This allows an issuer to transmit an attestation over insecure channels; even if a malicious party intercepts the attestation, the attacker cannot decrypt the information. The issuer also signs the attestation to ensure that its integrity is not compromised by a man-in-the-middle attack. Figure 2 presents the protocol for issuing attestations. Currently, we do not require the recipient to acknowledge receiving the attestation, although this could be added by a higher-level protocol.

### 3.3 Social Access Control Lists

In traditional ACLs, a list of identities specifies who is allowed access. A social ACL does not rely only on identities to specify access control. Instead, social ACLs can also allow access to others based on their social relationships. A social ACL contains the owner's public key, the public keys of all people who can access the object (like in traditional ACLs), and a social relationship. To access an object, people must either have their public key listed in the social ACL, or they must present an attestation issued to them by the owner certifying the relationship listed in the ACL. Social ACLs are signed to guarantee their integrity and authenticity. We also use XML to format social ACLs (see Figure 3).

Since not all social relationships are symmetric, the ordering in which public keys are listed in attestations and in social ACLs is

**Alice:** Generate the attestation  
**Alice:** Sign attestation with Alice's private key  
**Alice:** Encrypt signed attestation with Bob's public key  
**Alice:** Send encrypted attestation to Bob  
**Bob:** Decrypt received attestation with Bob's private key  
**Bob:** Check Alice's signature of the attestation  
**Bob:** Store attestation

**Figure 2:** *The protocol for issuing an attestation between Alice and Bob. The issuer and the recipient of an attestation know each other; they have each others' public keys in their address books before issuing the attestation. In this way, Alice can encrypt the attestation with Bob's public key, and Bob can verify Alice's signature.*

```

<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>
</access>
</acl>

```

**Figure 3:** *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.*

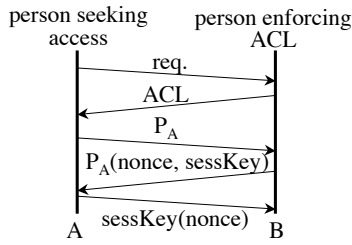
important. Using the terminology listed in Figures 1 and 3, Lockr grants access to an attestation if both the relationship type and the ordering of the identities (i.e., "firstParty" and "secondParty") match between the attestation and the ACL rule. However, as a convention, we always list the issuer's public key before the recipient's key if the relationship is between the issuer and the recipient. If the relationships involves the recipient and a third party, we list the recipient key before the other party's key. In this way, we can avoid the unfortunate case when an attestation is rejected by an ACL because identities are listed in the reverse order although the social relationship is symmetric.

#### 3.3.1 Enforcing access control with social ACLs

When requesting an object protected by a social ACL, the ACL enforcer returns the ACL (or the access rules found in the ACL at the minimum). Based on the ACL, the person seeking access determines which attestation to use to obtain access. When the ACL lists a conjunctive relationship, a person might have to send more than one attestation to obtain access. In this case, the attestations are concatenated in one single message. To access an object, a person must either have their public key listed in the ACL or an attestation certifying the social relationship listed in the ACL.

In the first case, the person must prove that they hold the private key corresponding to the public key listed in the ACL. For this, an ACL enforcer can issue a challenge (i.e., a nonce) encrypted with the person's public key. Access is granted only if the challenge is resolved. A session key is also setup to encrypt all subsequent communication. Note that no social attestations are exchanged. Figure 4 presents this protocol.

In the second case, the person must present a social attestation to access the object. Before transmitting the attestation, the person



**Figure 4:** *Checking the identity of a person seeking access. The ACL enforcer checks that the person’s public key appears in the ACL’s explicit list of users with access to the object. The nonce exchange checks that the person has the private key corresponding to the public key sent earlier.*

seeking access encrypts it with the attestation’s relationship key. In this way, only parties holding the same relationship key can decrypt this attestation. This ensures the privacy of attestations; without these relationship keys, users might be uncomfortable transmitting their attestation and, therefore, revealing their social relationships to others. A consequence of using relationship keys is that only parties with the same social relationship can decrypt attestations. In many scenarios, this limitation is convenient; for example, users can enforce that only their family members have access to family photos. However, in other scenarios, this can be too restrictive; for example, users might want Flickr to enforce their social ACLs restricting access to family members only. In these cases, object’s owners can choose to share their attestation’s relationship key with Flickr.

When sending an attestation, a person encrypts it with the relationship key. The party enforcing the ACL must also hold the relationship key to decrypt the attestation. To prove the identity of the attestation’s sender, a challenge is issued. If the sender resolves the challenge, access is granted. Session keys are also setup to encrypt all subsequent communication. Figure 5 illustrates this protocol.

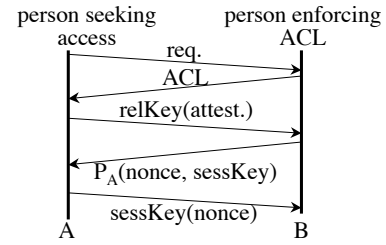
### 3.4 Attestation Revocation

We believe that people are often not concerned with revoking attestations that have not expired. Our attestations are used primarily for sharing content online; in many cases people will not bother restricting access to others just because their relationships changed. For example, when people share photos with their co-workers, they might not be concerned knowing their ACLs might allow access to former co-workers. The nature of the content does not warrant the need for revoking these attestations.

Nevertheless, there are scenarios when people would like to revoke certain attestations. We address these scenarios in three ways. First, attestations can be set to expire. Any system enforcing social ACLs will verify whether an attestation is still valid before granting access. Second, we augment ACLs with *exclusion lists*. The role of the exclusion list is to enumerate the people who cannot access the content even if they hold the appropriate attestation. For example, if Alice wants to share content with all her friends except Eve, Alice must add Eve’s public key to the social ACL’s exclusion list. Finally, a different way to restrict access is to reissue new attestations. The new attestation must have a new relationship key; in this way, Eve cannot access the content nor intercept and decrypt these attestations.

## 4. SECURITY PROPERTIES

The main assumption behind Lockr’s security properties is that people exchange their public keys along with their contact informa-



**Figure 5:** *Checking the attestation of a person seeking access. The attestation is encrypted with the attestation’s relationship key before sending it. The nonce exchange checks that the person seeking access has the private key corresponding to the public key of the attestation’s recipient.*

tion in a secure manner. If the public key of an individual listed in a personal address book is replaced with an attacker’s public key, the attacker can impersonate this individual. Based on this assumption, Lockr guarantees the following security properties:

**1. Integrity of attestations.** Since attestations are signed with the issuer’s private key, only the issuer can modify an attestation. This ensures that no attacker (including an attestation’s recipient) can modify the attestation’s content or forge an attestation.

**2. Non-transferability of attestations.** Even if attestations are passed to others, enforcing access control requires the authentication of the attestation’s holder as its recipient. This is done through a challenge that only the party holding the private key of the attestation’s recipient can solve. This also guarantees that access rights cannot be delegated (unlike with typical capabilities) – no one other than the issuer can grant access to the content. We believe that it is an appropriate choice for personal content sharing, since it enables the content owner to retain full control over the content.

**3. Privacy.** The attestations’ relationship key ensures that attestations can be revealed only to other key holders. If a relationship key is lost, stolen, or disclosed publicly by one of its holders (perhaps maliciously), then the privacy of all attestations sharing this key is compromised. However, having the attestation’s relationship key or even having a copy of the attestation is not enough to gain access. The person seeking access must also hold the attestation recipient’s private key. We believe that mounting this attack is difficult in practice since an attacker must also track down the set of susceptible attestations once the relationship key has been compromised.

**4. Resilience to man-in-the-middle attacks.** In our protocols, the person seeking access must authenticate, whereas the person enforcing the ACL does not. Moreover, once an attestation is transmitted, all communication is encrypted with a session key. This limits the damage of a man-in-the-middle attack to disrupting or serving corrupted content.

### 4.1 Attacks

Because attestations are encrypted with the relationship key when enforcing ACLs, an attacker must have access to this key to decipher the attestation. If any recipients of this attestation decide to make the relationship key public or to share it with an attacker, the privacy of all other recipients of this attestation can be compromised. Since the recipients of an attestation have a social relationship with the issuer, we believe that “social pressure” will act as a deterrent to these types of privacy attacks. In practice, this attack is analogous to one of a person’s friends trying to learn the identities of the other friends of this person.

Another possibility is that an attestation’s recipient decides to sell the attestation for monetary reward. Since the attestation is not



**Figure 6:** The LockrCenter interface for issuing and requesting attestations.

forgeable, the recipient must also sell their private key along with the attestation. In this case the attacker can use *all* the recipient’s attestations. This raises the cost that a recipient incurs by selling an attestation – the recipient relinquishes control over *all* their attestations, even the ones they will receive in the future. Note that the security of Lockr does not rely on the relationship key not being compromised. By compromising a relationship key, an adversary can only learn about the identities of parties involved in a relation, but the adversary cannot get access to the protected content.

## 5. APPLICATIONS

Lockr fits the needs of sharing personal content online. We believe that its simplicity and portability will motivate its adoption by Internet systems that deliver and share personal content. In this section, we describe how we added Lockr to two well-known Internet systems – BitTorrent and Flickr. First, we implemented LockrCenter – an attestation manager that allows people to issue and receive attestations. We integrated LockrCenter with Facebook to target a large user population (i.e., the Facebook users) that already shares personal content. Second, we extended Azureus, a popular BitTorrent client, to support our social access control scheme. Finally, we implemented Lockr for Flickr – a Firefox extension that allows users to control access to their content hosted on Flickr. We only describe our applications briefly due to lack of space.

### 5.1 LockrCenter: An Attestation Manager

An attestation manager has two roles – to allow people to exchange attestations, and to allow a user’s applications to retrieve attestations to gain access to protected content. The attestation manager can be implemented in many different ways, such as a standalone desktop application, an extension to an address book or an e-mail client, or even as an application running on a person’s mobile phone. Based on their preferences, people can use any of such attestation managers in the same way as people use many different e-mail clients or calendars.

We implemented LockrCenter as a Facebook application that allows users to issue and to request attestations. LockrCenter stores a user’s private key, public key, and the set of all received attestations in the user’s Facebook account. LockrCenter also allows any applications running on behalf of the user to retrieve the stored attestations. With LockrCenter, a person can issue an attestation to another by entering a Facebook userid or an e-mail address. If the other party is not using LockrCenter, an invitation e-mail to install

Social Torrent
<b>info:</b> meta-info about files in the torrent (file size, piece hashes, etc.)
<b>announce:</b> tracker announce URL
<b>announce-list:</b> list of backup trackers
<b>creation-date:</b> torrent creation date
<b>comment:</b> torrent owner’s comments
<b>created-by:</b> name and version of program used
<b>acl:</b> <acl>....</acl>
<b>signature:</b> {info, announce, owner, ACL, creation-date, created-by} <sub>owner</sub>

**Figure 7:** Social Torrent. A social torrent contains two additional key-value pairs: an ACL and a digital signature.

LockrCenter is sent. Figure 6 illustrates LockrCenter’s interface.

### 5.2 Integrating Lockr with BitTorrent

We integrated Lockr with BitTorrent by modifying a recent version of the popular BitTorrent client application Azureus (version 3.0.3.5). We tested our implementation on both Windows and Linux. At a high-level, we made two modifications to this Java-based client. First, we added social ACLs. Second, we added an attestation exchange and verification step during the formation of BitTorrent connections between peers.

#### 5.2.1 Social Torrents

We modified Azureus’s torrent creation module to support the creation of social torrents. A torrent file is simply a list of key-value pairs that describes the content served and the tracker coordinating the peers distributing the content. We added two key-value pairs to a torrent: a social ACL and a digital signature. We described the social ACL earlier in Section 3 (see Figure 3). The digital signature prevents a malicious party from tampering with the social torrent file (e.g., removing the social ACLs). Figure 7 illustrates the new fields added to a social torrent.

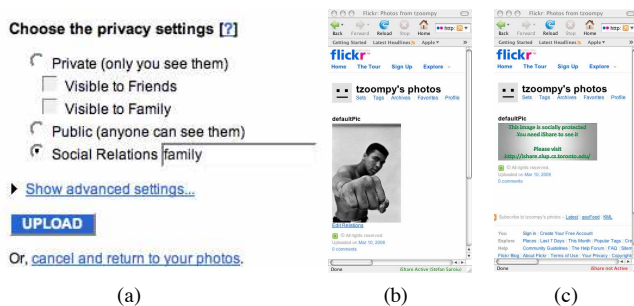
#### 5.2.2 Social Handshake

With Azureus, two peers perform two handshakes before exchanging data – a generic BitTorrent handshake and an Azureus-specific handshake. Our social handshake occurs immediately after these handshakes but before the data exchange. Peers engage in a social handshake only if they detect that their parties are running our modified version of Azureus. This is done through a special bit in a reserved field used by BitTorrent to extend its functionality. In this way, social Azureus is still backwards compatible – it can serve both social and regular torrents.

To implement the new handshake, we added three new messages to the Azureus messaging protocol [2] corresponding to the identity and attestation verification steps described in Section 3. If any of these checks fail, the connection between the peers is terminated immediately. The data exchange begins only after each peer proves its access to the content to the other.

### 5.3 Integrating Lockr with Flickr

We developed a Firefox extension (i.e., a browser plug-in) to add our access control scheme to Flickr. At a high-level, the plug-in performs two tasks. First, when a user uploads an image to Flickr, the plug-in extends the upload interface to add a social ACL. If the user chooses to restrict access to the image, the plug-in uploads a dummy place-holder image in addition to the original image to Flickr. Access to the dummy image is made unrestricted. In contrast, access to the original image is restricted with a social ACL;



**Figure 8: Lockr for Flickr.** In part (a), a user creates a social ACL. In part (b), a user views the protected image with the appropriate attestation. In part (c), a user without the appropriate attestation sees a dummy image.

we use a private server to manage and enforce access. The second task performed by the plug-in is using the attestations to obtain access to the protected content. When a user visits a page with Flickr photos, the plug-in queries the private server for social ACLs. Once the private server responds with the social ACLs, the plug-in sends back the attestation satisfying the ACL. Finally the private server replies with the secret URL of the real image stored on Flickr if the attestation provided is correct and the plug-in replaces the fake images with the real ones.

The plug-in performs this entire process in a manner transparent to the users. Figure 8 illustrates Lockr for Flickr’s interface. Figure 8a shows the interface of uploading a picture to Flickr. The user can make the content “private”, “public”, or protected with “social relationships”. The first two options are part of the default Flickr behavior, whereas our plug-in adds the last option. Figure 8b shows the original image when viewed by a user with appropriate attestations, while Figure 8c shows a generic image corresponding to a visit made by a user without the appropriate attestations (or without having the plug-in installed).

Despite lacking server support, our implementation of Lockr for Flickr is secure. No user can obtain access to photos protected with a social ACL unless having an appropriate attestation. Although all access to protected content is mediated by our own server, our server performs the same social ACL enforcement steps as any of the Flickr servers would perform in a Lockr implementation with server support from Flickr. By enforcing the social ACL, our server acts just an extension to the flickr.com domain.

Our plug-in is a short-term solution for adding our access control scheme to Flickr to demonstrate its practicality and ease-of-use. A much simpler and elegant implementation is one where Flickr offers server-side support for social ACLs and verification of social attestations.

## 6. RELATED WORK

Lockr’s social attestations and social ACLs are inspired from a number of pre-existing mechanisms that all attempt to provide authentication and access control in large-scale networked systems. In this section, we briefly present some of this previous work and we examine how Lockr relates to these systems.

**PGP** [7] On the surface, Lockr’s trust model appears similar to the one used by PGP – there is no centralized public-key infrastructure. However, there are two key differences between Lockr and PGP. First, PGP uses a vetting scheme in which people sign each other’s public keys. To verify a person’s signature in PGP, people must find a chain of trust linking the person to themselves. Over

time, PGP creates a “Web of trust” in which people accumulate each other’s signatures once verified. Instead, Lockr is centered on pre-existing social relationships and not on a single trust/no-trust relationship. In Lockr, people only certify a direct relationship with others. Some of these relationships might be very close (e.g., family), others could be distant (e.g., employees of a large organization). Unlike in PGP, these social relationships already exist in real-life, but they have yet to be captured digitally. Second, PGP’s trust relationship is transitive. However, since social relationships are diverse and complex in real-life, we have decided not to attempt to combine them to create transitive relationships.

**OpenID** [6] is a decentralized single sign-on system. Using OpenID-enabled web sites, the users do not need to remember multiple usernames across different Web sites. Instead, a user needs to be registered on a Web site with an OpenID “identity provider”. Since OpenID is decentralized, any Web site can employ OpenID software as a way for users to sign in. A user’s OpenID remains the same even when the user moves to a different provider. This simplifies access control as OpenID identifiers can be used globally. There are two important differences between OpenID and Lockr. First, in Lockr users do not need to trust third-party identity providers; instead, Lockr relies on social trust between people. One implication of this difference is that attestations are social networking scoped, whereas OpenID identifiers are global. Second, users rely on passwords to authenticate in OpenID making them vulnerable to phishing attacks [8]; instead, Lockr uses public key-based authentication in which users are challenged every time they authenticate or request access.

## 7. CONCLUSIONS

This paper presents Lockr – an access control scheme that makes sharing personal content easy. Lockr relies on two simple observations: (1) social relationships are a natural way to describe access control policies for personal content; and (2) people’s social networking information must be separated from content delivery and sharing. We described the design of Lockr, its security properties, and limitations. We also presented an implementation of Lockr in the context of two popular systems for sharing content online – BitTorrent and Flickr.

## 8. REFERENCES

- [1] Alexa. Alexa, The Web Information Company, 2008. <http://www.alexa.com>.
- [2] Azureus Messaging Protocol. Azureus Extended Messaging Protocol, 2008. [http://www.azureuswiki.com/index.php/Azureus\\_messaging\\_protocol](http://www.azureuswiki.com/index.php/Azureus_messaging_protocol).
- [3] Scott Garriss, Michael Kaminsky, Michael J. Freedman, Brad Karp, David Mazieres, and Haifeng Yu. Re: Reliable email. In *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, May 2006.
- [4] Roxana Geambasu, Magdalena Balazinska, Steven D. Gribble, and Henry M. Levy. Homeviews: Peer-to-peer middleware for personal data sharing applications. In *Proc. of SIGMOD International Conference on Management of Data*, Beijing, China, June 2007.
- [5] Peter J. Keleher, Neil Spring, and Bobby Bhattacharjee. Chit-based access control. Technical Report CS-TR-4878, University of Maryland at College Park, 2007.
- [6] OpenID. OpenID, 2008. <http://openid.net/>.
- [7] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. Wiley; 2nd edition, 1995.
- [8] Marco Slot. Beginner’s guide to openid phishing. <http://openid.marcoslot.net/>.