ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Access Control in a Peer-to-peer Social Network

Master's thesis submitted by

Youssef Afify

for a
Master of Science in
Communication Systems

ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE

supervised by:

Prof. Karl Aberer (EPFL)
Dr. Sonja Buchegger (Deutsche Telekom Laboratories)
Le Hung Vu (EPFL)

August 15th. 2008

Acknowledgments

I wish to express sincere thanks to Dr. Sonja Buchegger for her insight and guidance during this thesis work. I have also truly appreciated her help and welcomed tips in many matters of my 3 months stay in Berlin. My special thanks to Le Hung Vu for his relevant comments and suggestions for the evolution of the thesis. I would also like to thank Dr. Anwitaman Datta for his precious input during the weekly brainstorming we had about the project. Finally I would like to thank Prof. Karl Aberer for taking time from his busy supervise my thesis.

Abstract

It is no doubt that online social networks are currently the hottest trend in the web. During the last two years many web applications emerged to provide with their own system. However, a lack of innovation and a fierce concurrence have slowed the trend, and many services were left without other choice than to close down. Much like what happened with the business of Search Engines and the Google's merciless dominance, Facebook and Myspace are growing very fast. The centralized access and data control is a serious privacy problem, and a solution is to provide with a pure peer-to-peer infrastructure so that users get the control over their data. Many research questions on how to implement a functional Peer-to-peer Social Network (P2PSN) are raised in the fields of storage and security. In this thesis we concentrate on access control as it is one of the most challenging feature of centralized servers to distribute among an untrusted network of peers.

Contents

1

Online Social Networks

1.1 Definition

As of the day of writing, Wikipedia's entry for *Social Networks* lists more than 120 different services. Each service is distinguished by its built-in technology, the features it offers, and the clients gathered around it. The common vector is to rally communities of people to share and explore each others interests and activities. Users are encouraged to build a profile and share it with a list of other users with whom they may have a real life social bond, or just share a virtual connection. The first Online Social Networks (OSNs) were actually built as services for strangers to meet online, where as today, the successful websites are more or less a transposition of the real life social graph.

1.2 Features

Profile The profile is generated when subscribing and filling out the forms to access the SN service. This typically include a profile photo, and basic information such as age, location, education and work. It also contains a section about interests, such as favourite movies and books.

Friends and List of Friends The users are encouraged to build their virtual network by adding *Friends*, or *Contacts* depending how it is called on the service. A mutual agreement is usually required to set this relationship. A list of friends is also common and users can browse the list of their friends of friends to discover people they know. In order to extend the users social graph, Facebook has added a ”people you may know” feature that finds people within a few degrees or so of separation and suggests them as potential friends.

Public Messaging: *The Guestbook* It is the equivalent of the *wall* in Facebook, or the *scrapbook* in Orkut where your friends can post messages that will be readable by friends or everyone, depending on the social network's privacy settings. This is a popular feature, present in many of the existing applications. The "social interactions" between users become semi public.

Private Messaging In addition, SNs often have a private messaging functionality, similar to a web based e-mail. A study showed that younger generations prefer to use their social network service for sending messages, rather than using emails.

Media Sharing In SN such as Imeem, music video sharing is the main feature and the service is entirely built around it. But media sharing is also present as a complementary feature in other general purpose SN such as Facebook, where the shared videos are either embedded links to popular viral videos found on youtube, or user generated videos. The majority of social networks also come with a photo sharing facility, with the possibility to organize photos in albums, tag them, and post comments.

Status Update: Micro Blogging This short message allows users to broadcast their whereabouts, and moods or any other statement that they would like to share.

Groups, Communities, Events Groups can be used in two different ways. Either people gather around a common interest, or news event and share ideas, or visit the group to see upcoming events. Otherwise groups can be used just to show an affiliation to almost anything.

News Feeds It is a summary of the activities of the user's friends on the social network. This feature is a way of keeping up with what people are up to, like photo uploads, or public messages they exchanged.

Search engine Search and browsing options are often available not only for finding old friends or classmates, but also to discover events, publicly shared media, groups.

Applications, and Openness It's important to provide an open social network, and allow developers to add applications to increase the interactions, and give a better user experience.

The visibility and privacy settings Sites like MySpace allow users to choose whether they want their profile to be public or "Friends only". Since the notion of friendship is misleading in this context, this is clearly insufficient. Facebook is officially claiming that privacy is their number one concern. Recently they introduced granularity in the privacy settings, allowing users to create lists of users and define what each list can access. However, the default settings are set towards openness, and people of the same "network" can view each other's profiles without being friends, and many users are not familiar with the settings.

1.3 Privacy issues

The popularity of social networking has introduced a new way for creating, maintaining and developing relationships between individuals. This is leading to an increasing private information being stored and handled by the sites central servers. This data can be further processed to gather valuable marketing information, or propose undesirable targeted advertisement. Social networks are a window into the habits of a large panel of consumers and the commercial benefits of the sites are obvious.

On November 6 2008, Facebook introduced the Beacon system claiming that it will help users point out interesting products to each others and allow them to share information with their friends about their online shopping activities. The system was in reality intended to provide with a more profitable advertising tool by targeting userÕs buying habits mapped to their social graph. Beacon reported to Facebook when a user made a purchase on a Facebook partner website. For example, if a person buys a ring or a DVD on a partner site, Beacon broadcasted that to all the users having access to the buyer's profile. The problem is that Facebook didnÕt ask the users to opt in to the system. The default settings made publicly available data about shopping activities that were supposed to be private, such as christmas gifts. With all the buzz about this, Facebook finally introduced new privacy settings to give users more control over how Beacon works.

Social networking are particularly popular among teenagers, and parents often have little knowledge of their online activities and the information they share on public. Therefore, these networks attract a consequent number of sexual offenders. As a result law in the United States is adapting itself to protect teenagers by closely monitoring the activity on these networks with the collaboration of the sites.

1.4 Peer-to-peer Social Network

1.4.1 Why peer-to-peer?

Younger generation of social networks will be the first to have been exposed to social networks for a long period of time. The long terms effects on publishing one's life through MySpace and Facebook from a social point of view are not yet well known. Furthermore, online targeted advertising is becoming more and more aggressive and opaque, and the clients are not always aware of how their data is processes. If things go wrong in the future, and people awaken to the importance of privacy, their behavior can quickly change. Once privacy is challenged, there will be a greater sensitivity to information being provided to a single company.

Peer-to-peer seems like the natural solution in this situation. Distributing the various operations among the users, decentralizing administration and eliminating the notion of ownership of social networking systems instead of having a single company or institution processing users data, will give back ownership to users over their data, as long as the system provides with robust access control and security.

Now that we justified the likeliness of a future need in a new social networking technology, we may ask ourselves whether peer-to-peer matches the needs of this category of applications.

Social networking applications are in the heart of the Web 2.0 applications, stressing that end users are no longer simple consumers of content stored in central servers but are rather actively participating and publishing all sorts of reviews, medias or opinions. This questions the use of client/server paradigm in almost all existing web 2.0 applications, where as peer-to-peer may seem more appropriate. Clearly P2P could be the key to the future of social networking and online video for its scalability when it comes to storing large multimedia files.

One of the latest attempt in that sense was AllPeers, a firefox extension which provided with browser integrated file sharing and social networking features. Even if the service seemed very promising, it finally closed down. The reason given in their website was: "we have not achieved the kind of growth in our user base that our investors were expecting, and as a result we are not able to continue operating the service", in other words it was just for marketing reasons.

Additionally, ubiquitous connectivity with a market growth of Wifi mobile phones, and the success of mobile applications such as the ones for the iPhone are also an incentive to explore feasibility of a P2PSN. Many popular web applications are now available on mobile platforms. At the same time there is a huge potential for P2P mobile applications starting with instant messaging, IP telephony, video and other media sharing and location based services. We envision a p2p social networking platform as a basis for encompassing all these features, and offering ubiquitous connection and interaction between friends.

1.4.2 Challenges

If the centralization of the processing of private data by large companies becomes the leitmotiv for switching to a peer-to-peer social network, then a purely decentralized system is the first priority when devising a Peer-to-peer Social Network (P2PSN). However, in the current trend, the average user is far more concerned about his own circle of friends or family accessing unauthorized data than if Google is processing, deriving and storing statistics about him, his search queries or any other internet habit. Therefore, it is also primordial and crucial to provide with robust access control mechanisms, and a guarantee of a secure technology.

So access control is a central component to drag users towards adopting P2PSN. As in real life social networks, there is a degree in friendship relations. We want to be able to choose what we want to share and who we want to share it with. We also want to be able to revoke no longer desired content, with a mechanism for a quick and synchronized update of data.

A critical design problem comes from the granularity of the read and write operations. The goal of the project is not to devise new cryptographic tools and algorithms, but rather to use existing ones in the novel context of p2p social networks.

Social networks are highly dynamic networks. Distributed storage solutions tend to assume that nodes have a high expected session time (from the instant the node joins the system until it leaves). This is usually not the case in Social Networks, where users log in once or twice a day for a relatively brief moment. It seems that without this assumption a cooperative and persistent storage system will consume a huge amount of bandwidth overhead in relocation of data.

Users tend to check a small subset of their friends spaces. We can give them the choice of subscribing to the updates they want to receive. This should be taken into account in the storage policy, by caching for example objects close to high query nodes, with an intelligent mechanism automatically adapting to the users habits.

Facebook lingered before providing with flexibility in the privacy settings. The choice used to be limited to sharing with everyone, your network, your friends, or no one. Now among those old millions of users with each hundreds of friends, only few will take time to set these privacy rules taking into account each of the hundreds of already added friends. Too bad for Facebook, because even if most of people say they don't care about privacy, the user experience is better when the interaction is targeted only within a subset of friends you decided.

How to provide availability (immediate access to data when needed) and reliability (data persistence) of the p2p system as a whole? This is usually done by means of replication. But now we need to design a system that allow users to access and modify their profile, and read their friends profiles from any computer, not just their own device. Indeed, existing p2p applications were designed mainly for file sharing from a home access: you use your home computer as a download station. Now it is a dynamic system with frequent joins and departures, that can take into account that users may want to access their profile from different stations. Added to that, we can imagine that ensuring synchronization of a content which is meant to be updated among several nodes can create a problem of overhead traffic, mostly because of the nature of the content published by social network users (video and images). So even caching is hard to implement in a dynamic system.

In order to better understand the questions we are tackling in this thesis, we need to address what are the related p2p applications with some of the requirements of a P2PSN. Already existing solutions can then be used and modified to serve as a solution for our design.

2

Peer-to-peer Security and Access Control

2.1 P2P Technology

Peer-to-peer technology is the basis for new applications in distributed computing systems characterized by the direct sharing of resources such as CPU, storage, or content rather than through a centralized server. Scalability, self organization, resistance to censorship and absence of a central authority are many of the reasons why important research is carried out to find new applications based on this architecture.

In this section, we will define peer-to-peer technology, and focus on some popular content distribution mechanisms. Peer-to-peer systems can be classified into three categories : communication and collaboration, distributed computation, and content distribution.

2.1.1 Definition

Pure peer-to-peer systems refer to distributed systems where all nodes are equivalent in terms of their roles and tasks. The *Client/Server* communication is no longer applicable, as each node can act as both a server and a client. This restrictive definition does not take into account many applications and topologies that rely on some centralized infrastructure to conduct tasks as bootstrapping. An example is Kazaa [4], which is based on *superpeers*, special nodes that carry out the role of servers for a subset of the network. Shirky [9] defines peer-to-peer as a “class of applications that take advantage of resources, storage, cycles, content, human presence available at the edges of the internet”.

Many different applications have been implemented in a peer-to-peer fashion. Among the classification found in [11], the category of *Content Distribution* is the one that is of interest to us.

2.1.2 Content Distribution

A peer-to-peer content distribution system provides with a distributed storage facility. Users are able to *publish*, *search* and *retrieve* data objects.

File sharing is the most popular applications in this category. It provides with a facility for a one-off file exchange between peers. This best-effort approach does not address security issues and availability, and file sharing has been a vector of virus propagation. This along with many condemnation for copyright infringement contributed to the bad reputation of peer-to-peer technologies.

Peer-to-peer storage systems fall also in the category of content distribution, and are now very popular. They provide with a distributed storage medium where users store, and distribute content over the network. Security, access control and availability of data are crucial in these systems.

2.2 Security Principles

A lack of a centralized authority in p2p systems induces several challenges on the security of the designed systems. However, the scalability and availability properties of p2p file sharing has attracted research towards providing with a decentralized architecture for applications such as storage and file systems, content distribution and multicasting, or e-mail. Yet many improvements have to be made in order to guarantee the security of these systems in order to widely deploy them on a Internet scale. This will definitely revamp the reputation of the p2p paradigm, and release the pressure on ISP to block p2p traffic that suffers from the illegal file sharing.

Authentication and Authorization

Authentication is the process carried out by an entity to confirm the identity of another entity or to confirm that a data is indeed from whom it claims to be. Passwords, digital signatures or message authentication codes are the common techniques of authentication.
Authorization on the other hand, is the process of granting to the users some privileges on the access to a set of resources according to what is permitted to

them. The most popular techniques to enforce authorization are to maintain access control lists (ACL) listing the access rights of entity.

Availability

Availability is the proportion of time a system operates in normal conditions. In the case of peer-to-peer systems the challenge is to provide with a continuous access to data all the time. System failures and denial of service attacks (DoS) are very difficult to prevent, but through replication techniques it is possible to increase tolerance to malicious behavior and provide the users with a quick access to data whenever they request it. Replication techniques introduce a high storage cost and bandwidth overhead for maintaining consistency across replicas.

Confidentiality and Integrity

Confidentiality is reached when data is protected from unauthorized disclosures, where as Integrity is means that data is safe from unauthorized modifications. Encryption is a powerful guarantee of confidentiality, while data integrity can be achieved using digital signatures and message authentication codes. Replay attacks involve a malicious user injecting old data on the system. In order to guarantee freshness, timestamps or nonces can be used, but these tools require a certain degree of synchronization between the entities. Finally, and in order to ensure end-to-end security, communication channels with efficient security protocols must be set.

Key Management

When using cryptography based techniques to guarantee confidentiality, integrity, authenticity or access control in a peer-to-peer environment, users will soon be managing many keys. Therefore, efficient and scalable techniques must be designed, and special care must be taken when facing what can soon become very expensive operations. One such example is revocation of access rights which result naturally in re-encryption (decryption and encryption) of all the files and their replicas.
Key recovery mechanisms allow authorized persons to retrieve lost keys with the help of a set of trusted parties.

2.3 Threshold cryptography

The absence of centralized authority along with the dynamic characteristics of node presence necessitate the distribution of delicate operations over the peers, such as the ones usually handled by a Certification Authority (CA). Threshold Cryptography was introduced by Desmedt and Frankel [1]. It distributes the ability to provide a cryptographic service such as decryption or signing, over a number of parties exceeding a threshold. It increases the fault tolerance to the unavailability of some nodes as the task is still performed on their absence. This technique reduces the amount of trust placed on each entity, and the system no longer depends on the correct behavior of one single entity to perform the task in its entirety. Many applications on authentication and access control [3] can be jointly performed by a set of *delegates*.

Among the many schemes, we will be particularly interested in the ones providing with *distributed signature*.

The scheme provided by Shoup [10] is a reference. First define a k out of l threshold signature scheme is a protocol that allows any subset of k delegates out of l to generate a signature. However, the signature is not generated if less than k delegates participate in the protocol. k is called the *quorum*, where in law, "a quorum is the minimum number of members of a deliberative body necessary to conduct the business of that group"[Wikipedia].

The Byzantine General Problem

The Byzantine Generals Problem [6] is a computer science problem that deals with faulty behavior of nodes in a distributed system. The name of this problem comes from its abstract representation. A set of generals of the Byzantine Empire's army are deciding whether to attack an enemy or not. They are all in separate locations and can only communicate by sending to each others messengers. Some of the messengers are traitors and complicate the task of reaching a fair agreement by attempting to trick the process. A fair agreement is reached if and only if more than 2/3 of the messengers are acting loyally.

2.4 Related Work

Peer-to-peer storage and file systems are getting very popular. The growing unused disk space on user's desktops suggests that it can be used for replication to provide availability. At the same time, the decreasing computational cost of cryptographic operations encourages research on distributed security. However, several assumptions made by existing decentralized file systems are not compatible with the requirements of social networks.

One way to classify distributed storage systems is to distinguish between archival only and continuous update systems. In archival only systems like Freenet, on assumption is that any item stored is independent of any other item stored. Continuous update systems on the other hand are closer to file systems. They rely on a Distributed Object Location and Retrieval (DOLR) mechanism, and can provide with the technology for handling shared write operations, and maintaining a persistent relation between objects. Since objects are updatable, it is crucial to keep track of the latest version.

2.4.1 Farsite

Farsite [8] is a serverless, distributed and scalable file system. The intended work environment is within the desktop stations of large corporations or universities. These machines are interconnected with high bandwidth links and are running for the majority of the time. The read/write pattern is supposed to be sequential, and rarely concurrent. To achieve privacy, the techniques of encryption, one way hashing and raw replication are employed for large file data. On the other hand, Directory Metadata is small but must be accessible directly to the system. It is maintained by Byzantine replicated state machines and other cryptographic techniques to insure metadata syntax enforcement without compromising privacy. When a client writes a file, it encrypts the data using the public keys of all authorized readers of that file, and the encrypted file is replicated and distributed to a set of untrusted file hosts. The encryption prevents file hosts from reading files they are not granted access to, and the replication prevents file hosts from maliciously destroying a file. ALso, the maachines communicate with each other over cryptographically authenticated and secured channels, which are established using public-key cryptography. Therefore, each machine has its own public/private key pair (separate from the key pairs held by users), and each machine computes a large unique identifier for itself from a cryptographically strong hash of its public key. These machine identifiers are verifiable and cannot be forged as only the machine has the private key needed to sign the certificate. One of the key elements of FARSITE is to provide the benefits of Byzantine fault tolerance while avoiding the cost of full Byzantine agreement by using signed and dated certificates to cache the authorization granted through Byzantine operations.

2.4.2 Freenet

Freenet is a purely decentralized virtual file system focusing on security, publisher anonymity, deniability, and data replication for availability and performance. Files in Freenet are identified by unique binary keys obtained by applying a hash function. Each Freenet node maintains its own local data store, that it makes available to the network for reading and writing, as well as a dynamic routing table containing the addresses of other nodes and the files they are thought to hold. To search for a file, the user sends a request message specifying the key and a timeout value. In order to upload a new file in the network, a node sends a data insert message to itself with the binary key of the file. When a node receives the insert message, it first checks to see if the key is already taken. If not, the node looks up the closest key (in terms of lexicographic distance) in its routing table, and forwards the insert message to the corresponding node. This allows for files of similar keys to be stored at the same node. This automated process continues until the timeout is reached so that the file is replicated among several nodes. If no collision of keys happened, an all clear message will be sent back to the original inserter, otherwise the file with the same key will be sent back to the inserter as if he requested it. Now when a node receives a request for a file it has locally stored, the request ends with the node sending the file to the requester. Otherwise it forwards the request to its neighbor that is most likely to have the file, by searching for the file key in its local routing table that is closest to the one requested.
Although files are encrypted by a randomly generated encryption key, the confidentiality is not guaranteed. Indeed, the decryption key is stored along with the file's identifier, and any requester could read the file content. Actually, Freenet provides with anonymity through plausible deniability, where it is desirable for a typical user not to know what he is storing in his machine.

2.4.3 Plutus

Plutus [5] approach is to group objects with the same access rights into a filegroup, then only one key is needed to access all these files. Plutus opts for the so called lazy revokation introduced by Cepheus. Each time you want to cancel someone's right to read an object you must reencrypt the object (decrypt with the old key, then encrypt with a new one). You can also wait until the next update before doing so. This way, you will reduce overhead traffic but slightly affect the security: for unauthorized reading, the expelled user now only needs to keep a copy of the key instead of the whole file.
A technique called key rotation, or regression, avoids the reencryption of

all the files of a filegroup, when only one file is updated. This file alone is reencrypted with a new key K1 that is distributed to the members of the filegroup. K1 is generated such that everybody who knows K1 can derive the old key K0, but not reciprocally.

3

Access Control Mechanisms for P2PSN

3.1 P2PSN Jargon

In the following section, we will present the vocabulary that we choose to employ during the rest of this thesis in an attempt to between the changing names of features accros social networks.

One of the core concepts behind social networks is the notion of *Shared Space*. A shared space is characterized by a set of *members*, a set of data *objects*, and a persistent guestbook where people can post comments. A shared space can be a group, a network, or a member's personal space (profile). In the latter case, we define any user's space as a shared space with a set of members (his friends), a set of data objects (notes, albums, videos, Work and Education...), and a persistent guestbook. So for example A is friend of B if A is included in the set of members of B's space. Groups are also shared spaces, with members having different rights in reading, writing or deleting the data objects. As an example we can consider two types of groups: An Open Group, where anyone can join without invitation and where even non members can browse and read the content, but have to join to participate. On the other hand, an Invite Only Group would be invisible to non members, and as its name states requires an invitation to be joined. The invitation process is up to the Creator who can either reserve this right to himself or share it with a subset of members. This subset should be dynamic. For both groups there should be a set of default settings for the new joiners defined by the creator, and that would be easy to change later.

We can assume, without loss of generality that a shared space is owned by a unique user, usually the one who created the space: *The Owner*

3.2 System Overview

The access control mechanism is designed to be cryptography based and decentralized. However, certain operations are difficult to implement without trading off a certain amount of trust to a central authority. In order to limit this trust, the idea is to distribute the role of the central entity among a set of *delegates*. Through a Byzantine fault-tolerant protocol, along with an assumption on the maximum number of malicious nodes, it is possible to enforce the access control mechanism of this central authority in a distributed fashion.

The delegation techniques rely heavily on the behavior of the users involved in the process, where as cryptographic techniques, if well designed, can provide with information theoretically secure systems. Therefore, we choose here to maximize the operations based solely on cryptography, and adopt delegation as an ultimate solution. For instance, implementing decentralized access control for *multiple* writers on a *single* object demands delegation of authority for signing updates and maintaining consistent ordering of these updates across the replicas. This functionality is much more crucial in collaborative work environment than in Social Networks, and thus we assume that our system does not provide it. Indeed, the Owner of a Space in Social Networks uploads a content that is immutable (pictures, videos, notes...) justifying the assumption. However, there are still certain functions in social networks involving *single object/multiple writers*, and we will see later how the introduction of an object *bundle* will emulate them.

Some other key factors of the design are a sound choice of the delegates and efficient consensus protocols.

3.3 Filegroups

When the Owner of a Space wants to upload a file into the system, he must assign it to a *filegroup*, which is a set of objects having the same authorized readers. This is done by including the unique filegroup ID in the object header. In our case, an intuitive filegroup ID can be a hash of the combination between the owner's public key, and the actual filegroup name. For example : $Hash(PublicKey||FamilyAlbum)$. The public key can guarantee that the file group is unique in the system, and its combination with the filegroup's name provides with intuition when querying the files. Now each filegroup is associated with a *Key List Object*, a list containing the material to decrypt and read the objects of this filegroup. This Key List Object is signed with the owner's private key.

We justify the use of Key List objects because it is the best solution to provide with availability of credentials, independent of the device that a user is connected from. Another method would force users to rendez-vous, or physically exchange credentials, which will then have to be carried around wherever a user goes.

When creating a filegroup, the Owner generates an *asymmetric readers key pair*:

- PK_r will be known to the Owner only. It is used to encrypt objects that belong to the filegroup. Note that this key can eventually be shared with chosen friends to allow them to upload objects to the Owner's space. Each object must be signed with the private key of the user inserting it in order to authenticate it.

- PK_r^{-1} is the key for decrypting (reading) the objects. It is stored in a key list object.

3.4 Read Access Control

We have seen that the reading key PK_r^{-1} is stored in a Key List object. It remains to be seen how it is protected from unauthorized access.

The owner generates a symmetric key SK_i for each authorized reader i of the filegroup. Each of the symmetric keys are in turn encrypted with the corresponding reader's public key PK_i, and stored in the key list object. The filegroup's reading key PK_r^{-1} is then stored encrypted with the symmetric key SK_i in the object Key List:

Figure 3.1: Key List Object

GUID Reader i	**Encrypted Symmetric Key**	**Encrypted Readers Key**
GUID Reader 1	$\{SK_1\}_{PK_1}$	$\{PK_r^{-1}\}_{SK_1}$
GUID Reader 2	$\{SK_2\}_{PK_2}$	$\{PK_r^{-1}\}_{SK_2}$
...	...	...
GUID Reader N	$\{SK_N\}_{PK_N}$	$\{PK_r^{-1}\}_{SK_N}$

Why not store only the reading key PK_r^{-1} encrypted for each reader with his public key? This is mainly for a performance reason. The goal is to reduce the number of asymmetric key operations which are significantly more costly than symmetric operations. Particularly this will reveal important when facing with membership changes, as we will see later.

3.5 Write Access Control

As stated earlier in this write-up, it is best to avoid write operations as they require delegation of access rights for signing updated objects. However, we need to replicate an important feature: the *comments* posted by authorized friends of the Owner on the uploaded objects, or on the guestbook. These comments can be seen as *single object/multiple writers*. Instead of having one object *guestbook* that multiple writers update by appending their comments, we introduce here *the bundle*, a different kind of object, which purpose is to collect the objects of type *comments*, and display them together in a consistent and ordered fashion. Here are the steps followed by an authorized writer w_i:

Step 1 w_i creates a message m_i

Step 2 w_i retrieves the key PK_r stored in the bundle's corresponding key list object

Step 3 w_i encrypts the message m_i with PK_r and sign it with his private key $PK_{w_i}^{-1}$

Step 4 w_i stores $\{m_i\}_{PK_r} || \{H(\{m_i\}_{PK_r})\}_{PK_{w_i}^{-1}}$ in the object store

Step 5 w_i sends a request to the *delegates* to append his object to the *bundle*

Step 6 The *delegates* sign the newly updated bundle

with:

- PK_i^{-1} : Private Key of user i.
- m_i : the message of user i
- $\{m_i\}_K = M_i$: message m_i encrypted with key K
- M_i : encrypted message m_i
- $||$: concatenation
- $M = \{m_1\}_K || \{m_2\}_K || ... || \{m_n\}_K = M_1 || ... || M_n$: the data part of object *bundle*
- $H(.)$: Secure hash function such as SHA-1

- $\{H(M)\}_K$: Signature

Remark: We will se later the details of **Step 5** and **Step 6**, and the case where w_i is not authorized to write.

Figure 3.2: *bundle*

$\{m_1\}_{PK_r}\Vert\{H(M_1)\}_{PK_1^{-1}}$	
$\{m_2\}_{PK_r}\Vert\{H(M_2)\}_{PK_2^{-1}}$	
$\{m_3\}_{PK_r}\Vert\{H(M_3)\}_{PK_3^{-1}}$	*Signature*

So each of these *comments* m_i are owned by their originator, and the Owner of the space where they are posted decides whether or not to display them according to a Write Access List. This list contains the public keys of the users allowed to write the object. When the Owner of the space is offline, the *delegates* are in charge of the access control on behalf of him. They are required to add the new message to the bundle and mutually sign it subsequently to verifying that a "wannabe writer" is indeed authorized to post the message on the e.g. *guestbook*.

3.6 Delegation Mechanisms

The main role of the *delegates* is to generate in a distributed manner a valid signature for the new bundle. In this section we will investigate in more details their functions and we will see how their role goes further than gate-keeping as they can actually offer authentication mechanisms and a freshness guarantee of objects.

3.6.1 Initiation

The first step carried out by an Owner is the *initiation* where a Write Access List is created containing the public keys of the authorized writers. In the case of social networks, a user having read access to a guestbook is usually also authorized to post his comments. In other words we barely need to discriminate readers from writers as a user is either both, or neither (none?). So in practice, and instead of having a Write Access List, we can use the Key

List Object for both reading and writing the comments. Recall that the Key List Object has an entry with the GUID of each of the authorized readers, and thus writers. However, we will keep on calling it the Write Access List just to underline the possibility of parting readers from writers. This may actually be useful when creating a space with only a subset of expert members generating content for a larger audience.

3.6.2 Assigning the delegates

The Owner chooses a set of delegates that will be responsible for the bundles related to a particular filegroup. Note here that it is better to choose the *number* of delegates always *larger* than actual *quorum* needed to run the consensus protocol. The reason for that is to tolerate faulty delegates that become no longer available, without having to appoint new ones. We can also envision the set of delegates as organizing itself to proactively accommodate and enroll new peers when facing leaving delegates, or even expel the ones that demonstrate signs of maliciousness by failing to agree with the majority, but that remains to be detailed in a future work.

Now assigning the delegates means giving to each one of them a share d_i of the signing key d. Delegates do not have the decryption keys to read the objects of the filegroup. Their role is limited to verifying that a potential writer is authorized, and eventually execute a protocol resulting in a distributed signature generation on the bundle. This protocol is based on Threshold Cryptography (TC) and shared signature schemes.

When choosing the right protocol, we must take into account the possibility of some delegates becoming unavailable. This means that the share of a no longer reachable delegate must be reconstructed and given to a new delegate.

Distributed Key generation for ad hoc groups operating without a trusted dealer for RSA based TC schemes require the generation of a product of two primes, and schemes [7] have been proposed to operate this in a distributed manner, however they require expensive overhead and are not applicable here since the owner is responsible of this operation. Threshold cryptography algorithms are analyzed in [3] with regards to access control in ad hoc groups.

3.6.3 Joining and Leaving delegates

The group of delegates will definitely evolve and an assumption on a never changing set of delegates is not reasonable. Threshold cryptographic schemes have been derived that are suitable for ad hoc groups in absence of a trusted dealer. Since the owner operates as the trusted dealer, the ad hoc operating mode is temporary.

3.7 Membership revocation

Membership revocation is the process of canceling all or a part of the access rights of a user. According to our scheme, it is obvious that when it comes to canceling a user's right to write , the owner of the space only needs to update the Write Access List. Then the delegates when running their distributed signature protocol will fail to map the expelled user to this List, and abort signing the object.

Now things get more complicated when it comes to dealing with revocation of read access. The simple scheme that comes to mind is straight forward. When the owner decides to expel a user, he first starts by decrypting the objects that he no longer wants the former reader to access, then re-encrypt these objects with a new key. Then the final step is to update the Key List Object accordingly. This already seems to be a very heavy operation, particularly, we must note that these operations of Decryption/Re-encryption have to be applied to *ALL* replicas of *each* concerned object that are in the network. If we choose this costly method, then at least it must be for the sake of guaranteeing that the objects will no longer be accessed in the future by the expelled user. A full and synchronized cooperation with the overlay and storage layers are crucial for enhancing this method. The owner might as well do everything locally by encrypting (assume he has his files locally stored in his personal device, and are thus not ciphers) with the new key the objects concerned by the membership change. Then he proceeds to a new *store request* as he does for new uploaded objects, and updates the credentials on the Key List Objects. It remains to be seen how to deal with the *garbage* files that are still available on the network encrypted with the old key. Indeed, a delete operation is not always sufficient to get rid of every single version and replica of the old file. Particularly sometimes the network suffers from partitioning or maybe some pieces of data, if not the whole file, are stored in a computer which hasn't logged in for a while and could have missed the *delete* operation for some reason.

So clearly if this method does not guarantee that once all the heavy operations of decryption/re-encryption are done no former reader will still be able to read, then it is not worth spending computation in it. Moreover, no one can prevent a reader to archive the files he is accessing. So by simply storing the files locally when he had access to them, an expelled reader make all the efforts of protecting this data vain.

The concept of *lazy revocation* invented by Kevin Fu [2] is very interesting in our design. The basic idea is also used in Plutus [5], and consist in using a new encryption key only for updated objects or newly added ones preventing the owner from all the overhead of re-encrypting every file. The owner will first modify the Key List Object by deleting the entry of the expelled user, which will actually be no longer able to read old files unless he kept a copy of the key stored locally. Compared to the first design, this decreases the security because earlier he had to keep the whole file, now the key alone is sufficient. In a normal setting where users barely keep everything stored this is not a major problem.

Conclusion and Future Work

P2P social networks are an opportunity to leverage research in many topics. Each of the different features is the occasion to further extend the knowledge in a specific area of distributed systems: Event notification and News Feed would fit into a Publish/Subscribe paradigm, where as p2p email may be enhanced to serve the needs of P2PSN email like private messaging. Additionally, P2PSN can benefit from the active research in distributed storage systems. In this work we have tried to gather some of the existing techniques used in decentralized access control for distributed environments and apply them in the novel context of social networking.

File sharing applications have known a tremendous success thanks to p2p, and are still very popular. Nevertheless, when it comes to providing with more complicated, feature-rich and very secure systems, pure p2p technology has not shown great success yet. Even if scalability and cheap maintenance are the major attractions, it remains to be seen how to operate efficiently under malicious behavior without any centralized component.

Moreover, p2p applications require the installation of a software, and people are still unwilling to do so which led to some applications switching to web based model such as iMeem. The real opportunity remains in mobile telephony, where web applications are quickly developing as shown by the 60 millions applications sold by itunes for the iPhone. With the fierce competition in this domain, along with the users unwillingness to shift from their usual application to adopt a new one, this often resulted in many system shutting down (Allpeers), or never expanding (Skyrider).

Furthermore, it is clear that the pressure from the music and movies industry adds further hurdles in the expansion of p2p applications.

Finally, online social networking has found an important place in people's life, and the trend is growing, so we can envision an exciting future for these services. Research on data portability and openID are promising directions forming the basis of a *social web*. This is certainly an add up to user experience, but can quickly become a privacy nightmare if data is not protected from unauthorized access and manipulation.

Bibliography

[1] Yvo G. Desmedt and Yair Frankel. Threshold cryptosystems. In *CRYPTO '89: Proceedings on Advances in cryptology*, pages 307–315, New York, NY, USA, 1989. Springer-Verlag New York, Inc.

[2] Kevin Fu. Group sharing and random access in cryptographic storage file systems. Master's thesis, Massachusetts Institute of Technology, 1999.

[3] Nitesh Saxena Gene Tsudik Jeong Hyun Yi. Threshold cryptography in p2p and manets: The case of access control. *Elsevier*, 2007.

[4] Kazaa. web site: http://www.kazaa.com. *P2P File Sharing Application*, 2003.

[5] Mahesh Kallahalla Erik Riedel Ram Swaminathan Qian Wang Kevin Fu. Plutus: Scalable secure file sharing on untrusted storage. *Proceedings of the 2nd Conference on File and Storage Technologies (FASTÕ03)*, 2003.

[6] M. Pease L. Lamport, R. Shostak. The byzantine general problem. *ACM Transactions on ProgrammingLanguages and Systems*, 1982.

[7] Philip D. Mackenzie and Moti Yung. Robust efficient distributed rsa-key generation. In *a*, pages 663–672. ACM Press, 1998.

[8] Atul Adya William J. Bolosky Miguel Castro Gerald Cermak Ronnie Chaiken John R. Douceur Jon Howell Jacob R. Lorch Marvin Theimer and Roger P. Wattenhofer. Federated, available, and reliable storage for an incompletely trusted environment. *5th Symposium on Operating Systems Design and Implementation*, 2002.

[9] Shirky. What is p2p... and what it isn't. *O'Reilly*, 2000.

[10] Victor Shoup. *Practical Threshold Signatures.* EUROCRYPT, IBM Zurich Research Lab, Switzerland, 2000.

[11] Stephanos Androustellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys, Vol. 36, No. 4, December 2004, pp. 335371*, 2004.