COMP208 - Group Software Project
Ballmer Peak

Choi, S.F; M. Chadwick; P. Duff; L. Prince; A.Senin; L. Thomas

February 9, 2014

Contents

1

Project Proposal

1.1 Introduction

The proposed project is a simple, privacy oriented social network, which demands zero security knowledge on behalf of its users. In order to ensure security and privacy in the face of nation state adversaries the system must be unable spy on its users even if it wants to or server operators are ordered to.

We feel that obscuring the content of messages isn't enough, because suspician may, and often does, fall upon people not for what they say, but to whom they are speaking. Our system will therefore not merely hide the content of messages, but the recipient of messages too. Hiding the fact that an IP address sent a message is out of scope, but hiding which user/keypair did so is in scope. citation needed

The system will provide the following functionality:

- A user may add friends
- A user may IM with fellow users
- A user may IM anonymously with fellow users
- A user may post messages to all their friends on their wall (think FB)
- A user may request to post messages to all of a friends friends on a friends page (think FB wall, permission is required because a user cannot know who their friend (or anyone) is friends with)

The server operator will have access to the following information:

- Which IP uploaded which message (although they will be ignorant of its content)
- Which IPs are connecting to the server as clients
- What times a specific IP connects [1]

[1] While this will aid in tying an IP address to a person, it is deemed acceptable because it is not useful information unless the persons private key is compromised.

Talk about TLS, end-to-end crypto

A third party logging all traffic between all clients and a server will have access to the following information:

- Who connects to the server, whether they upload or downloand information [2]

The benefits we feel this system provides over current solutions are the following:

- Server operators can not know who talks with whom
- Server operators can not know the content of messages
- Server operators can not know which message is intended for which user
- Server operators can not know who is friends with whom
- Third parties sniffing the connections can not know anything the server operator cannot know (this isn't unique, but is worth stating).

In order to ensure nobody can tell who is talking with whom we will base our security model on the idea of shared mailboxes, as seen in practice at alt.anonymous.messages [3]. In this model one posts a message by encryping it using the public key of the recipient, and posting it in a public location. In this model one reads a message by downloading all messages from that location, and attempting to decrypt them all using ones private key. Our protocol will build atop this simple premise, and the the server will be a mere repository of messages, the real work occuring wholly in the client.

1.2 Summary of protocol

pretty dataflow diagrams

Creating an account is done by generating an RSA keypair, and choosing a name. An unencrypted (but signed) message is then posted to the server associating that keypair with that name. In this way, by knowing the public key of someone, you may discover their name in the service, but not vice versa.

Connecting for the first time Every unencrypted message stored on the server is downloaded(signed nicknames and nothing more) [4] (if someone retroactivly grants you permission to view something they publish it as a new message with an old timestamp). At this time the local database contains only signed messages claiming usernames. The public keys are not provided, these are of use only when you learn the public key behind a name. The rationale for not providing public keys is provided in the section regarding adding a friend. Messages posted after your name was claimed will require downloading too, as once

[2] size correlation attacks could be used here if the message content is known
[3] https://groups.google.com/forum/#!forum/alt.anonymous.messages
[4] clients use bittorrent to lighten server load?

you claim a name people may send you messages.

Connecting subsequently The client requests every message from the last time they connected (sent by the client, not stored by the server) up to the present. Decryptable messages are used to update the local DB, others are discarded.

Continued connection During a session the client requests updates from the server every 1-5 seconds (configurable by the user).

Adding a friend is performed by having a friend email (or otherwise transfere) you their public key. This is input to the client, and it finds their name (via public posting that occured when registering). You may now interact with that person. They may not interact with you until they recieve your public key.[5]

Talking with a friend or posting on your wall is achieved by writing a message, signing it with your private key, and encrypting one copy of it with each of the recipiants public keys before posting it to the server. The client prevents one from posting a message to someones public key, if they have not claimed a nickname.

Posting to a friends wall may be requested by sending a specially formated message to that friend (all handled by the GUI, like much else here), when that friend logs in they will recieve your request to post on their wall and may confirm or deny it. If they confirm then they take your (signed) message and transmit it to each of their friends as previously described (authentication is entirely based on crypto signatures, so it doesn't matter who posts the message).[6]

1.3 Tasks and Challanges

What would need to be done:

- Server
  - stores messages sent to it by clients
  - sends stored messages to clients at request (either the whole lot, or messages from within a certain timeframe)
- Client
  - handle decryption/encryption

[5]This is the one part that will be difficult for normal users, however any protocol by which the server stores and serves public keys is entirely unsuitable as a MitM would be trivial on behalf of the server operators

[6]This is required because it is impossible for one to know who their friends friends are.

- sends data to the server
- maintains a local DB built from decrpyted messages
- displays a nice GUI that hides everything from the user

1.3.1 Data Flow to Clients and Required Server Storage

A estimate is hereafter given as to the size of all stored messages, and the amount of data which would need downloading by each client when it is started. The following assumptions are used:

- A users avarage message posted to their wall is 200 characters
- A users avarage number of messages posted to their wall per day is 10
- A users avarage number of friends is 100 (each and every friend represents one key exchange)
- A users avarage private message (to single user) is 50 characters
- A users avarage number of private (to single user) messages per day is 300

With these generous estimates, each user would generate (200*10*100)+(50*300*1) bytes of raw data per day. Assuming a 10% protocol overhead we would see 236,500 bytes of data per day per user.

The storage space required for a server is therfore 86MB per year per user. On a server with 50,000 users that has been running for 3 years, there would be just 1.3TB of data.

Every time a client connects, it must download all messages posted since it last connected to the server. To mitigate this we may run as a daemon on linux, or a background process in windows, that starts when the user logs in. If we can expect a computer to be turned on for just 4 hours a day then 20 hours of data must be downloaded. ((236,500*no_of_users)/24)*hours_off_per_day bytes must be downloaded when the users computer is turned on.

The following table shows the delays between the computer turning on, and every message having been downloaded (assuming a download speed of 500KB/second, and a netowrk of 1000 users).

To mitigate this, posts will be downloaded in reverse order, so that more recent posts are downloaded first. We feel that waiting 2-5 mins is an acceptable delay for the degree of privacy provided. Once the user is synced after turning their computer on, no further delays will be incurred until the computer is shut down.

Due to the inherantly limited network size (<1500 users of one server is practical) we recomend a number of smaller servers, each serving either a geographic location, or a specific interest group.

While this latency could be avoided, and huge networks (>1,000,000) used, it would come at the cost of the server operator being able to learn that somebody

Hours off per day	Minutes to sync
0	0
4	1.3
10	3.2
12	3.9
16	5.2
20	6.5

Table 1.1: Hours a computer is turned off per day vs minutes to sync

is sending or recieving messages, and also who those messages are sent to/from (although they couldn't know what the messages said).

1.4 Task Partitioning

- D. Breslin
  - Task 1
  - Task 3
- L. Thomas
  - Task 2
  - Task 4

Replace task partitioning placeholders

1.5 Proposed Tools

Git, with a central (private) repository hosted on github will be used for version control. The language used has yet to be decided...

choose programming language

Talk more about tools, specifically about communication between group members

1.6 Deliverables

- Protocol documentation for developers writing third party clients
- Windows and Linux executable: client
- Windows and Linux executable: server
- Full source for server, client, and any associated works

update with names of associated works as project continues

2

Requirements

2.1 Anticipated Software

Description of anticipated software

- ensure optimal security, when passing messages, viewing profile
- ensure server will not able to detect any activies between the users of the system
- users can share posts only to specific people
- users to keep their own data and only friends with users as well
- users is to pass public key with any type of medium they prefer to other users

The purpose of this system is to emphasise on the security measures of a social media by encrypting user details such as profile information, messages which are used to pass to other users, and posts which are only designated for specific users t[o read?]

this was missing from the original commit too

The system is to have strict security measures implemented. It is able to encrypt messages with the use of RSA and AES. The only way for the other user to decrypt the data is if it was encrypted using their public key; which is given from the recipient to the sender via whichever medium he prefers, e.g. email.

Server capabilities are strongly limited as it only funtion is to handle the traffic of the encrypted data between different users.

Appendices

Appendix A

Deadlines

- **2014-01-31** topic and team
- **2014-02-14** requirements
- **2014-03-14** design
- **2014-05-09** portfolio & individual submission

Appendix B

Licence

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean auctor sapien est, nec porttitor massa iaculis vel. Curabitur ac elit et velit laoreet euismod a id ante. Suspendisse potenti. Maecenas mattis risus id diam eleifend dictum. Nunc cursus tempor pharetra. Donec luctus dolor imperdiet, tristique sapien gravida, facilisis dui. Integer eget ornare lorem, sit amet porta tellus. Suspendisse eu arcu orci. Donec non lectus non odio sagittis elementum. In non adipiscing purus, at vehicula turpis. Proin eu iaculis libero, quis vestibulum lorem. Etiam nisi lorem, pellentesque nec ante in, consectetur varius erat. Maecenas elementum semper orci ac iaculis. Donec eu molestie mauris, non hendrerit magna. Proin pretium nec nisi tincidunt facilisis.

Choose a licence

Nullam in pharetra libero, quis eleifend sem. Nunc porta vestibulum risus non tempor. Phasellus vestibulum ullamcorper eros. Vivamus venenatis elit ut ligula porttitor tempus. Maecenas pellentesque pellentesque neque. Sed eros sapien, eleifend et egestas at, interdum sit amet lorem. Mauris leo quam, semper eu velit vitae, rhoncus blandit nunc. In libero ante, blandit at sapien eget, cursus dapibus dui. Mauris vestibulum urna at elementum ultrices. Curabitur dictum felis at ultricies accumsan. Maecenas ullamcorper scelerisque leo, eget luctus ipsum. Vivamus pretium neque eget quam convallis viverra. Proin ac tristique eros, bibendum laoreet ipsum. Fusce condimentum nisl placerat tortor cursus, sit amet commodo leo porttitor.

Donec pharetra accumsan est ut dapibus. Cras pharetra, augue a facilisis rhoncus, sem nisi pretium massa, id vestibulum turpis mauris eleifend lacus. Quisque tincidunt tellus felis, sit amet eleifend quam porttitor vitae. Integer sagittis dapibus turpis, tempus pharetra libero condimentum sed. Pellentesque nec volutpat nulla, ut molestie diam. Pellentesque accumsan, ligula ut commodo cursus, sapien erat faucibus arcu, in viverra nunc augue ac turpis. Phasellus ultricies urna eget sollicitudin mollis. Vivamus justo metus, cursus ac ipsum sed, fermentum faucibus tellus. Morbi commodo tempor ipsum at pretium. Aenean vitae orci lacinia, dapibus mauris vel, auctor metus. Etiam gravida rhoncus enim. Suspendisse ligula erat, ullamcorper et orci quis, sagittis semper ante.

Todo list