

COMP208 - Group Software Project  
Group X

D. Breslin, L. Thomas, et. al.

January 28, 2014

# Contents

<b>1</b>	<b>Project Proposal</b>	<b>3</b>
1.1	Introduction . . . . .	3
1.2	Summary of protocol . . . . .	4
1.3	Tasks and Challenges . . . . .	5
1.3.1	Data Flow to Clients and Required Server Storage . . . .	6
1.4	Task Partitioning . . . . .	6
1.5	Proposed Tools . . . . .	6
1.6	Deliverables . . . . .	7
 <b>Appendices</b>		
<b>A</b>	<b>Deadlines</b>	<b>9</b>
<b>B</b>	<b>Licence</b>	<b>10</b>
	<b>Todo list</b>	<b>11</b>

# 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 suspicion 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 <sup>1</sup>

---

<sup>1</sup>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 download information <sup>2</sup>

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 <sup>3</sup>. In this model one posts a message by encrypting 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 one's private key. Our protocol will build atop this simple premise, and the server will be a mere repository of messages, the real work occurring 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) <sup>4</sup> (if someone retroactively 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

<sup>2</sup>size correlation attacks could be used here if the message content is known

<sup>3</sup><https://groups.google.com/forum/#!forum/alt.anonymous.messages>

<sup>4</sup>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 transfer) you their public key. This is input to the system, and it finds their name (via public posting that occurred when registering). You may now interact with that person. They may not interact with you until they receive your public key.<sup>5</sup>

**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 recipients public keys before posting it to the server. The client prevents one from posing 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 formatted message to that friend (all handled by the GUI, like much else here), when that friend logs in they will receive 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).<sup>6</sup>

## 1.3 Tasks and Challenges

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

---

<sup>5</sup>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

<sup>6</sup>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 decrypted messages
- displays a nice GUI that hides everything from the user

### 1.3.1 Data Flow to Clients and Required Server Storage

An 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 user's average message posted to their wall is 200 characters
- A user's average number of messages posted to their wall per day is 24
- A user's average number of friends is 200 (each and every friend represents one key exchange)
- A user's average private message (to single user) is 50 characters
- A user's average number of private (to single user) messages per day is 300

With these generous estimates, each user would generate  $(200 \times 24 \times 200) + (50 \times 300 \times 1)$  bytes of raw data per day. Assuming a 10% protocol overhead we would see 1,072,500 bytes of data per day per user.

server storage

The storage space required for a server is therefore

blah blah

Assuming each user opens their client just once every other day. To mitigate this we may run as a daemon on linux, or a background process in windows, that starts when the user logs in.

analyse this method w/r/t  
how often the computer is  
turned on

## 1.4 Task Partitioning

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

Replace task partitioning  
placeholders

## 1.5 Proposed Tools

choose programming lan-  
guage

Git, with a central (private) repository hosted on github will be used for version control. The language used will be fortran on punchcards.

Talk more about tools,  
specifically about commu-  
nication 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

# 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

Choose a 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.

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

citation needed . . . . .	3
Talk about TLS, end-to-end crypto . . . . .	4
pretty dataflow diagrams . . . . .	4
server storage . . . . .	6
blah blah . . . . .	6
analyse this method w/r/t how often the computer is turned on . . . . .	6
Replace task partitioning placeholders . . . . .	6
choose programming language . . . . .	6
Talk more about tools, specifically about communication between group members . . . . .	6
update with names of associated works as project continues . . . . .	7
Choose a licence . . . . .	10
Full author list . . . . .	11
Replace "Group X" with actual group name/code . . . . .	11

Full author list

Replace "Group X" with actual group name/code