COMP208 - Group Software Project
Ballmer Peak

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

February 9, 2014

Contents

Part I

Project Proposal

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]

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

Talk about TLS, end-to-end crypto

[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.

- 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.

[2] size correlation attacks could be used here if the message content is known
[3] https://groups.google.com/forum/#!forum/alt.anonymous.messages

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) [1] (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 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.[2]

[1] clients use bittorrent to lighten server load?

[2] 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

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).[3]

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

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

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).

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

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

4

Task Partitioning

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

Replace task partitioning placeholders

5

Proposed Tools

choose programming language

Talk more about tools, specifically about communication between group members

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

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

Part II

Requirements

7

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?]

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.

this was missing from the original commit too

8

Case Study: Facebook

8.1 User

this text is about what the user account can do , and what it need to resgiter the user account, and what to do if user forgot the password.

Different beteew user and non user This section is about what user funtion is doing in facebook , the retrsiger have to be done before the user want to use the funtion.

the funtioncan that non-user can use:

- search friends whose in facebook;
- can be able to check on the user's post(if the user allows the non-user to visit)

the funtioncan that user can use:

- Posting on wall
- Friends list
- Account Manage
- Live chating with friends

user Name register(signing up) non-User have to sign up some presonal information to become a user,The information can be use to reset the possword,Or display for other user (depends on have the user set the information as public) .

the information have given:

- email
- SureName
- LastName

- UserName
- Password
- irthday
- Sex
- user have to agree the terms and data Use Policy, including there cookie use.

User login in

The user can login in by given following information on www.facebook.com or apps that connect to facebook :

- e-mail or phone number
- Password

there might be some checking to the user. For example ,Once the user login in, the facebook AI will check on the userIP , if the IP shows the location is to far from last login IP , the facebook might need to confirm the user by asking whats the friend's name on the photo that user had tap .Nevertheless, If the user forgotten the password,ask for Email, Phone, Username or Full Name to help the user to get the password

8.2 Post

facebook allows user post text, video, image on there wall or friend's wall(if they are the facebook user.). This section is to clearify what the user can post and what the other user can do on what the user post.

type of post

- Message Posting
- picture Posting
- video posting
- Message and Picture posting
- video and Message posting

8.2.1 Post funtion(text Only)

post manage (poster)

- delete the post
- rewrite the post

- setting privacy(who can read the post)
- cancel comment funtion

post manage(reader)

- set Noticfication
- hide the post
- unfollow the poster

post contain

- poster Name(the user who post the information)
- the text from user post
- the video and image (not must be)
- sharing funtion
- comment funtion
- like funtion

8.2.2 Post funtion(image and video Only)

post manage (poster)

- delete the post
- rewrite the message
- setting privacy(who can read the post,who can tag the post)
- cancel comment funtion
- tag the image

post manage(reader)

- set Noticfication
- hide the post
- unfollow the poster
- report the image or video to facebook team (for example:the post is offensive)

post contain

- poster Name(the user who post the information)

- the text from user post
- the video and image (not must be)
- sharing funtion
- comment funtion
- like funtion

8.2.3 sharing

Firstly confirm is he user really wants to share than get in to the share function, the post will post on the user's wall and refance whose the origian user. **share function**

- allow user post other user's post to his friend (by posting to the users wall)
- allow user to share the post on third-party web(e.g. YouTube, steam information)
- user can input message which to sharing type post

8.2.4 comment

is a funtion that other user to comment the post.the following funtion is what poster and reader can do on commend.

comment function

- user can write comment on the post.
- user can like the comment.
- user can hide the comment.
- user can report the comment(e.g. sexuallt explicit content...).
- User can give feed back to the user who give comment.
- comment can be block(reader only).

8.2.5 Like

the user can like the post , and the name of the user who like the post will be on the list who liked the post.

- user can like on the post.
- user can like on the shared post.
- user can like the the comment.
- user can cancel the like .

8.3 Friend

the sever will store the inforamtion of which user is friend which other user by a simple processe .

the process to bcoming friend:

- the userA have to sent a friend request to userB.
- UserB have to accept the requset .
- user A and user B have friend relationship now .

after the process, the userA and UserB have both of there name stored in firned list. **Funtion after become friend**

- friend's post will update on News Feed.
- friends will be added in Chat funtion.

8.4 Personal Information

User can manage there account , what presonal information they want to show to public . all the presonal information, that user wants to show will contain on "About". User have option to hide all the information .

following information will contain in "About"

- Work and education
- PLece Lived
- Relationship
- Basic Information
  - Birthday
  - Relaionship
  - Status
  - Anniversary
  - Languages
  - Religious
  - Political
  - Family
  - Contact Information
- photo
  - all the photo have user's taged

- Friends
  - what friend user had
- Note
  - What Notes user droped and uploaded to facebook .
- Groups
  - What group have user join.
- Events
  - what Events user have.
- Likes
  - what page(unknow type) user liked.
- Apps
  - what apps user have.
- Books
  - what books page user liked/follow.
- TV programmes
  - what TV page user liked/follow.
- Films
  - What Films page user Liked/follow.
- Music
  - what Music(or stars) user liked/ follow.
- Sports
  - -what sport page user liked.
- Place
  - -where's the place that user had been .

8.5 wall

users wall

New feeds

User walls(TimeLine(facebook.ver)) Wall store all the post of the user posted since the account was created and the informaation about the user. other user can view the user's wall by clicking the name of the user. Other user can post on friends wall as well.In this case,both can deleted the post . raither than that, the only one can dlete the post is the user or manage.sorted by time.

wall contain : (middle part) all the post that friends posted to user all the post that friends mantion user's name* all the post the user posted user profile picture change

(left hand side) About Photo Friend Place Sport Music

right hand side ad(can cancel by Ad block) time line

8.6 Chat System

allows user to chat with friends , theres different state of friends shows on the screen, and the system will record the time of when user offline, and have the user read the message .

state of friends :

- online
- offline
- using moblie login

funtion of user can do in chat room:

- add more friends to chat room
- video calling
- sending files
- sending image
- mute conversation
- clear window
- report as spam link

Setting of chat system:

- Chat Sounds on/off
- user can allows spicecify friends see his online
- user can choose the state as "offline"

9

User Requirements

consider implementing a WOT system with levels of trust

turn all these lists into real text

9.1 Registration

Users may register by sending a CLAIM message to the server, this will claim a username for that user, and allow people they send messages to to see their username.

Before registering the user must generate an RSA keypair, they will be given the option of generating a new keypair, or using an existing keypair. The keypair provided will be encrypted using AES with the users password being used to derive the key. The user therefore must enter their password to log in to the client. The database will be encrypted using the same AES key as the keys are encrypted with.

9.2 Interacting with other users

People are adding by adding their public key, this is transmitted outside of our system, via whichever channel the users deem appropriate[1].

Adding someone is asymmetric. Just because you add them doesn't mean they've added you. You do not require consent to add someone, just their public key.

The system allows the user to manage their list of known people into groups such as friends, family, and cooworkers. The user defines these groups as lists of people whose public key they know. The user may create any group they desire, these groups are visible to only the user, and private.

groups should be posted to the server as a message only that user can read, this supports the same user using multiple clients (on, say, a phone and laptop)

[1] This is required to prevent server operators from MitM'ing users

9.3 Profile Data

Profile data will be transmitted via PDATA messages. Different versions of profile information may be provided to different groups of people. Profile data may be update by the user.

The supported fields in a PDATA message are:

- Name
- Username (unique, but this uniqueness is ensured by server and shouldn't be relied on)
- Birthday
- Sex
- E-Mail
- About

9.4 Account recovery

Account recovery is not possible without your keypair, due to the the GUI should urge the user to keep a copy on a flash drive, or external hard drive. The keys themselved will be encrypted with the users password.

9.5 Posts

9.5.1 Walls

Each user has their own wall. On their wall they may posts messages for themselves and others to see. All wall posts should be addressed to the user themself so they can see their own posts, otherwise they will be unable to even view their own posts. When posting to their wall they choose who will be able to see the post, whether this is a group or people, a specific list of people, or just themselves is up to the user. They will not however be given the option to post publically. Users may also post to another users wall.

Wall posts may contain links to other content, however this content is never thumbnailed[2].

A user may edit their old posts, however older versions will still be available for viewing; similarly users may 'delete' posts, but they are still visible to malicious clients.

Due to bandwidth limitations on such networks as we are building, a user may only post plaintext, they may not post images, video, or audio.

[2]client MUST NEVER thumbnail link or otherwise access it without EXPLICIT user consent (see tor/js exploit on freedom hosting by the USA and tracking techniques recently thwarted by GMail caching images

9.5.2 Commenting

All wall posts may be commented on by any user who can see them. Comments are visible to all people who can see the original post; due to this comments must be forwarded by original posters client to all the same recipients, as the commenter may not know whom the original posters allowed to see the post.

9.5.3 Liking

Any wall post may be liked. Likes are simply a specially formatted comment which contains only the text: "__LIKE__". As such they are handled in the same way.

inband metadata is probably a bad idea

9.5.4 Events

The client will alert the user to other users birthdays by automatically posting a wall post that only the user may read, which alerts the user of the event. These are normal wall posts.

9.6 Chat

Users may chat in real time, however messages can still be sent when one user logs off, to be recieved when they log in. Past conversations are saved, and a user may block users from messaging them; the client actually just ignores their messages, it's impossible to stop someone from messaging you.

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.

Appendix C

TODO

C.1 General

Errors shouldn't just display a message, they should be properly handled Get a real DB
REVOKE claims and messages after a certain date if private key leaked

C.2 Requirements

(Week 1-2) 1. Project Desc.

- Project being done for (Peter)
- Mission Statement (Luke)
- Mission Objective (Luke)

2. Statement of Deliverables

- Desc. of anticipated documentation (Luke)
- Desc. of anticipated software (Aishah)
- Desc. of any anticipated experiments + blackbox (Louis)
- Desc. of methods of evalutation of the work (Louis)
- System boundary diagram (Leon)
- User view and requirements (Luke)
- System requirements (Luke)
- Transaction requirements (Aishah)

3. Project and Plan

- Facebook research (Leon)
- Data required (???)
- Implementation Stage (???)
- Gantt Chart (Mike)
- Pert Chart (Mike)
- Risk Assessment (Mike)

4. Bibliography

- (Luke)

C.3 Code

NetworkConnection

- Runs in its own thread

GUI

- Use the database to display a user interface
- Display users walls

Parser

- Wall posts

Database

- Set up a database, none currently exists

Appendix D

Bugs

- The 'DB' allows adding a friend multiple times, no reason to fix because the whole thing needs rewriting as a real DB anyway

Todo list