COMP208 - Group Software Project
Ballmer Peak

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

February 10, 2014

Contents

Part I

Requirements

1

Mission Statement

The proposed project(Turtlenet) is a simple, privacy oriented social network, which demands zero security or technical 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 intend 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, as is which IP/user/keypair received the message and the content of the message.

citation needed

While there exist many tools for hiding what you are saying, relativly few seek to hide who talks with whom, and those which do oftenimplement it merely as a proxy.

this REALLY needs citations

We feel that these tools have significant usability problems, as was recently made starkly clear when Glenn Greenwald, a reporter for the guardian, was unable to work with Edward Snowden because he found gpg to be "too annoying" to use.

section comparing extant security systems such as Tor (talk about ssl-strip on exit notes, netowrk visibility, and node distribution)

"Its really annoying and complicated, the [email] encryption software" - Glenn Greenwald [5]

In light of dissidents utilizing social networking websites such as Facebook and Twitter to organize protests, we feel that there is a need for an easy to use, encrypted communications platform with support for real-time and asynchronus

communication between users.

2

Mission Objectives

Our goal is to produce an easy to use social network, within which nearly all data is encrypted.

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. We will also allow users to transmit public keys as QR codes, for ease of use.

The system will provide a platform for people to securely communicate, both one-to-one and in groups. Users will be able to post information to all of their friends, or a subset of them as well as sharing links and discussing matters of interest.

The following are our main design goals, please note that the system is designed with axiom that the server operators are evil, seeking to spy on every user, and able to modify the source for the server.

- strong cryptography protecting the content of messages
- make it an impossible task to derive, from the information the server has or is able to collect, which users send a message to which users
- make it an impossible task to derive, from the information the server has or is able to collect, which users recieve a message at all
- transmission of public key is easy, and doesn't require knowing what a public key is

- be intuitive and easy to use, prompting the user when required
- provide a rich social network experiance, so as to draw regular members and drive up network diversity

The server operator will have access to the following information:

- Which IP uploaded which message (although they will be ignorant of its content, type, recipient, and sender)
- Which IPs are connecting to the server as clients (but not what they view, , whom they talk with, or whether they recieve a message at all)
- What times a specific IP connects [1]

A third party logging all traffic between all clients and a server will have access to what IPs connect to the server, and whether they upload or downloand information [2]

Talk about TLS, end-to-end crypto

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

- 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

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 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 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 occurring wholly in the client.

[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.

[2] size correlation attacks could be used here if the message content is known

[3] https://groups.google.com/forum/#!forum/alt.anonymous.messages

3

Project Target

A project of this scope has a rather specific target in sight. Due to its encrypted nature, Turtlenet can act as a form of anonymity between users who would otherwise be targeted by governments and/or institutions opposed to them. Countries such as China and a majority of the middle east have recently seen negative press due to their persecution of individuals whom disagree with the ruling regime, such software would allow said individuals safety from what the wider world views as acceptable.

Large multinational defence corporations such as IBM, Thales, or BAE might also find Turtlenet useful, as it would allow for a secure communication tool between employees in an office. It could also potentially be used outside a company firewall to send messages securely between offices across much larger distances. Corporations such as defence contractors often hold security in the highest regard, and such a client would match their needs well.

A more likely recipient of this system however, is the internet itself, as we have decided to release Turtlenet under an open source license. Should another group decide to embark on a similar project, they will have access to this project, to act as a baseline for their own work. More on this license will be covered later in the portfolio.

4

Anticipated Software

We anticipate the creation of the following software:

- Windows, Linux, and OSX executable: client
- Windows, Linux, and OSX executable: server
- Windows, Linux, and OSX executable: installer for client and server
- Full source for server, client, and any associated works

update with names of associated works as project continues

The client will create and use an SQLite database, local to each client, this database will be used to store all information that the specific client is aware of.

5

Anticipated Documentation

We will provide the following documentation:

- Installation guide for server
- User manual for client
- Full protocol documentation for third parties wishing to implement their own clients
- Full description of system design and architechture, for future maintainance
- Full description of database design
- Interface documentation

Javadoc makes the code messy as fuck, just use LaTeX and render as HTML

6

Anticipated Experiments

6.1 Performance Testing

How well does the system response match the users work practice?

6.2 Robustness Testing

System level black box testing.

- Devise a series of inputs and expected outputs.
- Run these inputs through the system and record the actual outputs.
- Compare the actual outputs with the expected outputs.

Inputs used should range from expected use patterns to silly as users tend to do things totally unexpected.

6.3 Recoverability Testing

How good is the system at recovering from user errors?

6.4 Learnability Testing

How long does it take a new user to become productive with the system?

6.5 Security Testing

Are users able to view messages intended for other people?

7

Methods of Evaluation

...

8

Case Study: Facebook

8.1 Overview

A user has a profile with information about them, they may add other users as 'friends', friends may view each others 'posts' and talk to each other. Posts are multimedia messages typically visible to all the friends of the person who made the post. Most posts can be commented upon, and both posts and comments may be 'liked'. Liking merely publically marks the fact that you approve of something.

8.2 Registration

In order to be a user of facebook, one must register. In doing so you provide facebook with the following information, this may also be used to later reset the password of your account, should you forget it.

- First Name
- Last Name
- E-Mail
- Password
- Birthday
- Sex

friends of a user are automatically alerted of friends BD's

In order to register one must read and agree to their terms [4], read their data use policy [3], and read their cookie policy [2]. Given profile information can be changed at a later date, within certain bounds. Facebook requires the use of your real name, and in fact forbids all false personal information, under their terms.[**4.1**, 4]

8.3 Account Managment

The user is given the ability to set the security defaults for their posts and information. Tese options include who is able to see wall posts, whether comments are enabled by default, and who may see which aspects of your profile information. You can also manage the permissions granted to facebook apps.

more information on FB apps

A users profile may contain the following information:

- Work and education
- PLece Lived
- Relationship
- Basic Information
  - Birthday
  - Relaionship
  - Status
  - Anniversary
  - Languages
  - Religious
  - Political
  - Family
  - Contact Information
-

field	description
photo	all the photo have user's taged
friend	what friend the user had
note	what notes the 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 .

Table 8.1: user adds a new post

8.4 Friend

In facebook, 'friending' someone is symmetric; that is, if you are friends with them, they are friends with you. The facebook severs store which user is friends with which other users. Adding another user as a friend is simply a matter of sending that user a friend request, and having it approved by the second user. A user may see a list of all who are their 'friend' on FB, in the friend list. After friending somebody that persons wall posts will appear on your news feed, and you will be able to chat with that user.

In order to add friends, facebook allows you to see your friends friend lists, and search by name, email, and location for other users. Facebook also suggests other users whom you may already know IRL, based on your friends friends. Non-users are also able to search facebook for people that they may know.

8.5 Post

8.5.1 Posts, and functions thereof

Facebook allows a user to post on their wall or friend's wall (if they are friens with the facebook user). Posts may contain: text, images, videos, or any combination thereof.

A user posting a post may do the following:

- delete their own post

- rewrite their own post
- decide who may view a post, the options are as follows:
  - public
  - private
  - only-me
  - friends only
  - friends of friends
  - ...

does FB allow sharing to one or two specific people?

8.5.2 Interaction with anothers posts

A post will typically be displayed on the newsfeeds of the people who are able to see it, due to this the name of the person who made a post is always displayed next to it. Posts themselves may be commented upon, liked, and reposted to the viewers wall ('shared') with an additional message; the number and names of people who have liked a post is displayed underneath it; likes may be cancelled at a later date. The comment function may however be disabled by the user who makes a post.

A user may hide specific posts, or hide all posts by a specific user. They may also, instead of hiding anothers posts alltogether, merely prevent them from being automatically displayed on their newsfeed. A user may report an image, video or comment to facebook team (for example:the post is offensive). Comments may also be liked, hidden, and reported; following such a report FB is able to remove offensive or illegal posts.

See 72d5e2dc, what is 'set a notification'?

Images which are posted may be tagged, this allows other users to mouse-over parts of the image and be informed who is pictured. This functionality is also used to add all posted images of someone to their profile.

allow user to share the post on third-party web(e.g. YouTube, Steam information): really? I don't remember seeing this option in steam

8.6 wall

A users wall stores all the posts of the user posted since the account was created and the informaation about the user, this information is presented in reverse choronlogical order, so that recent events are at the top of the page and easily visible. Other users may view the users wall by clicking the name of the user from anywhere in facebook. Other users may post on a friends wall as well as

the owner, see section on posts for more information; In this case, both the poster and the ownder of the wall can delete the post. Facebook also retains the power to erase any content on its service.

Posts mentioning a user are automatically reposted to that users wall, this can occur manually or when that person is tagged in an image.

8.7 Chat System

Facebook allows a user to chat with their friends, and will inform a user of whether their friends are online or not (though this can be faked), and whether the user you are chatting with has read the last message that you sent them. Facebook determines that you have read a message when... . You are also informed whether your friend is logged in on a mobile device or not.

how does FB do this?

Whole groups of users may chat together, in multi-user conversations. Facebook also supports video calling and file transfer during chats. If a user does not wish to be bothered by another using chatting with them, then they may 'mute' that users conversion. Users spamming via chat may be reported to facebook. Because multi-user conversations (and indeed long running one-to-one conversations) can get rather large, facebook allows you to hide the history of a conversation.

Facebook chat alerts the user to new messages in a conversation by playing a sound.

8.8 Architechture

...

8.9 Security

In order to use facebook post registration a user must 'log in'. This places an authentication cookie on the users computer which gives anyone in possesion of it the ability to act as that user.Users login using their email and password.

verify they didn't change this since I left FB

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

9

User View

The user will be presented with a simple and easy to use interface, which assumes and requires no knowledge of security. The most complicated thing that the user will have to do is transmit to other users their public key. We plan too alleviate this process by encoding the public key as both a QR code and plaintext string (depending on user preference), both of which may be easily transmitted via email, SMS, meeting in person, or over any other channel.

Upon connecting to the system for the first time, the user will be prompted to enter a username, and any profile information they choose to share. They will be urged to avoid using their real name as their username, and informed that profile information is shared on a case by case basis, and is not automatically visible to people whom they add.

They will then be brought to the main page of the system, where they (and) people they authorize, may post message. There will be a prompt for them to add peoples public keys, and the option to add either an image or plaintext public key.

Upon adding anothers public key, they will first be informed of that persons username, and prompted to catagorize the person. The user will be able to create a number of catagories into which they can place that user. Already created categories will be displayed. One person may be added to multiple categories, and nobody but the user is aware that this occurs. Depending upon the categories the person is entered into, that person gains the ability to view certain content posted by the user.

When the user posts a message they are prompted to enter a recipient, this may be: a previously created category (such as friend, coworker); a number of

individuals; or any combination thereof.

Upon recieving a message a sound is played and the user is informed. They are then able to click on the notification to open the message, and chat. When chatting with another user they have the ability to 'ignore' that user, in this case the user will see no more messages from that user.

10

User Requirements

consider implementing a WOT system with levels of trust

turn all these lists into real text

10.1 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.

10.2 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]. We will provide a user with their public key as a QR code, or a plaintext string, depedning on user preference.

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

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

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)

10.3 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

10.4 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.

10.5 Posts

10.5.1 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.

10.5.2 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.

10.5.3 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

10.5.4 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.

10.6 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

[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

user may block users from messaging them; the client actually just ignores their messages, it's impossible to stop someone from messaging you.

11

System Requirements

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 11.1: 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).

12

Required Data

...

13

Transaction Requirements

- the transactions involved for each user activity
- how these data are used
- 3 categories
  - data entry
  - data update and deletion
  - data queries
- transaction should be related to user view, ensure all functions are supported

13.1 data entry

field	notes
username	user is to make his own username
name	user is to enter his name (first and last name)
birthday	user enters his date of birth by selecting the date from a calendar
sex	user is to select either his/her sex, male or female
email	user is to enter his email address

Table 13.1: User enters his profile information

user_id's and post_id's are local and don't exist outside the DB

field	notes
message_id	id is to be incremented when a new message is initiated
from(user_id)	system is to insert the user_id of whoever initiated the message
to(user_id)	user is to select the person whom he wants to send the message to
content	content of the message which the user intends to send to the receiver
message_times_date	the time and date is recorded of which the message is sent to the receiver

Table 13.2: User starts a new conversation by adding a new message

field	notes
post_id	id is to be incremented when a new post is added
permission_allowed_to	user is to choose specific users (he knows) to view his post
from	system is to insert the creator's name of the post
to	the user_name of the post is inserted if the user directs this post to specific pe
comment_id	comment_id lists down the comment made out from other users
content	the content of the post
message_time_date	the time and date is recorded of which the post is created

Table 13.3: user adds a new post

field	notes
event_id	id is to be incremented when a new event is added
title	title of the event
content	content of the event
from	the user_id of the person who posted the event
permission_allowed_to	user is to choose specific users (he knows) to view his event

Table 13.4: user adds a new event

field	notes
comment_id	
post_id	
comment_from	
comment_time_ date	

Table 13.5: user adds a comment

field	notes
like_id	
post_id	
like_from	

Table 13.6: user likes a post

14

Risk Assessment

14.1 Parallel Tasks

[1] A big concern for any project is the amount of tasks that will be performed simultaneously. For every task that is carried out together, but potentially separate from each other, risk is increased - with more tasks making a more dramatic increase of potential failure for the project. For the planning section of this project we have performed a large amount of tasks simultaneously which may be detrimental to our quality of work later in the project.

In order to reduce or even eliminate the risk of too many parallel activities, the project should be planned using Gantt and PERT charts to reduce the amount of tasks being performed simultaneously and have more milestones within the project. This will help effectively split up the project into more manageable sections which will not only make the project seem simpler to complete but will improve the monitoring capabilities of the project as well.

14.2 Group Work

Working within a group can make deliverable dates difficult to achieve. This can be due to a lack of communication, unavailability of party members or an incapability to meet deadlines for some of the members. A meeting of minds also includes an assemblage of work ethics. Because of this, work may grind to a halt as members argue over personal yet trivial matters such as formatting

documents or a varying opinion on what is classed as 'enough work' for a task.

Combating the disadvantages of working as a group can be difficult. As some problems are part of a group unable to function either properly of efficiently together, this can be the breaking factor of the project. This is a risk that cannot be eliminated but can be reduced. A way of minimising the amount of damage that the risk will do would be to have a centralised form of contacting members of the group - examples being a website or using a revision control system such as 'Apache Subversion 'or 'Git', will give a common area for the group to look for potential absences or reasons for reduce work output from members.

The best way to reduce the risk of differing qualities of work between the group would be to define a standard of work between the group - such as the layout of source files in programming languages or a house style for formal documents as part of the group's external identity. Having this be available to the group in some form, such as in a text file within a shared area will allow the group to refresh their memories of parts of the set standard that they wouldn't follow otherwise.

14.3 Deadlines

Deadlines are the final day or dates that an object needs to be completed by. Sometimes within a project the deadline may be overstepped due to any of the risks mentioned within this document, which can lead to something small such as being berated by the project leader or something serious such as a breach in contract with the client. For these reasons, deadlines need to be adhered to so that the project can continue on schedule.

Reducing the risk of deadlines are important, especially for those that are not capable of monitoring their time effectively. By providing deadlines as a range of dates as opposed to a singular date, there is increased flexibility within the project and it gives some people more time to finish their work if it is required. By using a range of dates the group can finish on the beginning of the deadline range - a sort of pseudo-deadline - meet up and discuss whether alterations need to be made on the work and then use the remainder of the time until the end of the deadline range to perform them.

14.4 Scope

15

Implementation Stage and Planning

15.1 Critical Path

gantt and pert chart, along with discussion of their value and applicability

15.2 System Boundary Diagram

descrpition of diagram, and why it is useful

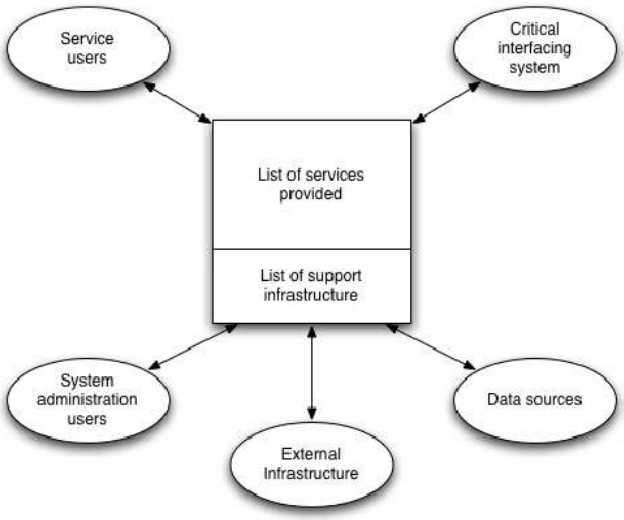

Figure 15.1: System Boundary Diagram

Part II

Design

16

Protocol

16.1 High Level 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.

[1] clients use bittorrent to lighten server load?

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]

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]

[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

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

17

Database

NB: Public keys are 217 characters long, all id's are auto-incremented.

attribute	description
id **PK**	
username	
name	
birthday	
sex	
e-mail	
public_key	

Table 17.1: table: users

attribute	description
id **PK**	
user_id **FK**	
name	

Table 17.2: table: category

attribute	description
id **PK**	
permission_allowed_to **FK**	this list of users are permissible to view the post, its comments and likes
from **FK**	
to **FK**	this can be NULL if the wall is not posted for a specific person
comment_id	
content	
time	

Table 17.3: table: wall_post

attribute	description
id **PK**	
login_time	
logout_time	

Table 17.4: table: login_logout_log

attribute	description
message_id **PK**	
from **FK**	
to **FK**	
content	
time	

Table 17.5: table: private_message

attribute	description
id **PK**	
post_id **FK**	from wall_post table
comment_from	
comment_time	

Table 17.6: table: comment

attribute	description
id **PK**	
post_id **FK**	
like_from **FK**	

Table 17.7: table: like

attribute	description
id **PK**	
title	
content	
from **FK**	
permission_allowed_to **FK**	

Table 17.8: table: events

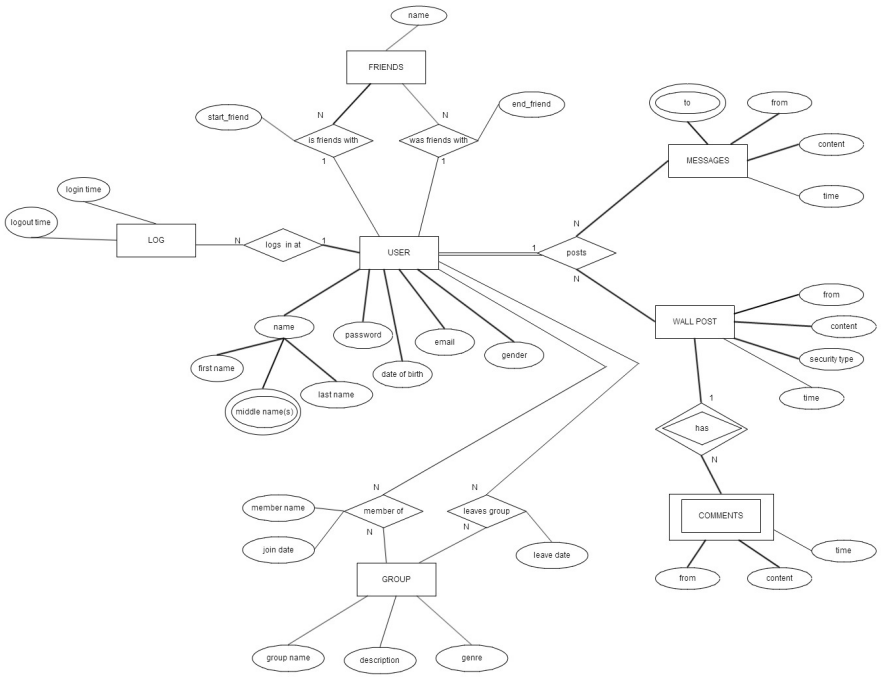

Figure 17.1: Database E-R Diagram

18

Transaction details

- the transactions involved for each user activity
- how these data are used
- 3 categories
  - data entry
  - data update and deletion
  - data queries
- transaction should be related to user view, ensure all functions are supported

18.1 data entry

field	notes
username	user is to make his own username
name	user is to enter his name (first and last name)
birthday	user enters his date of birth by selecting the date from a calendar
sex	user is to select either his/her sex, male or female
email	user is to enter his email address

Table 18.1: User enters his profile information

user_id's and post_id's are local and don't exist outside the DB

field	notes
message_id	id is to be incremented when a new message is initiated
from(user_id)	system is to insert the user_id of whoever initiated the message
to(user_id)	user is to select the person whom he wants to send the message to
content	content of the message which the user intends to send to the receiver
message_times_date	the time and date is recorded of which the message is sent to the receiver

Table 18.2: User starts a new conversation by adding a new message

field	notes
post_id	id is to be incremented when a new post is added
permission_allowed_to	user is to choose specific users (he knows) to view his post
from	system is to insert the creator's name of the post
to	the user_name of the post is inserted if the user directs this post to specific pe
comment_id	comment_id lists down the comment made out from other users
content	the content of the post
message_time_date	the time and date is recorded of which the post is created

Table 18.3: user adds a new post

field	notes
event_id	id is to be incremented when a new event is added
title	title of the event
content	content of the event
from	the user_id of the person who posted the event
permission_allowed_to	user is to choose specific users (he knows) to view his event

Table 18.4: user adds a new event

field	notes
comment_id	
post_id	
comment_from	
comment_time_ date	

Table 18.5: user adds a comment

field	notes
like_id	
post_id	
like_from	

Table 18.6: user likes a post

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.

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.

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

2. Statement of Deliverables

- **DRAFTED** Desc. of anticipated documentation (Luke)
- **DRAFTED** Desc. of anticipated software (Aishah)
- **INCOMPLETE** Desc. of any anticipated experiments + blackbox (Louis)
- **INCOMPLETE** Desc. of methods of evalutation of the work (Louis)
- **INCOMPLETE** System boundary diagram (Leon)

- **PARTIALLY DRAFTED** User view and requirements (Luke)
- **DRAFTED** System requirements (Luke)
- **INCOMPLETE** Transaction requirements (Aishah)

3. Project and Plan

- **ROUGHLY DRAFTED** Facebook research (Leon)
- **WTF IS THIS** Data required (???)
- **INCOMPLETE** Implementation Stage (???)
- **INCOMPLETE** Milestone Identification (Milestones can most easily be recognised as deliverables) (Mike)
- **INCOMPLETE** Gantt Chart (Mike)
- **INCOMPLETE** Pert Chart (Mike)
- **INCOMPLETE** Risk Assessment (Mike)

4. Bibliography

- **CITE MORE** (Luke)

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

Bibliography

[1] Oracle Corporation. *The Benefits of Risk Assessment on Projects, Portfolios, and Businesses*. English. White Paper. Enterprise Software, Computer Software, 2009. 14 pp. URL: `http://www.oracle.com/us/042743.pdf`.

[2] Facebook. *Cookies, Pixels & Similar Technologies*. https://www.facebook.com/help/cookies. 2014.

[3] Facebook. *Facebook Datause Policy*. https://www.facebook.com/about/privacy. 2013.

[4] Facebook. *Statement of Rights and Responsibilities*. https://www.facebook.com/legal/terms. 2013.

[5] Peter Maass. *How Laura Poitras Helped Snowden Spill His Secrets*. http://www.nytimes.com/2013/08/18/magazine/laura-poitras-snowden.html?smid=pl-share. 2013.