COMP208 - Group Software Project
Ballmer Peak

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

February 12, 2014

Contents

Part I

Requirements

1

Mission Statement

"to shift societal norms to a state wherein privacy is respected without caveat or justification"

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 asynchronous communication between users.

2

Mission Objectives

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

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

> "Its really annoying and complicated, the [email] encryption software" - Glenn Greenwald [11]

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

this REALLY needs citations

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

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

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 receive 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 experience, 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 receive 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 download

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.

information [2]

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.

[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

Threat Model

When designing a system in which security is a significant aspect, it helps to define clearly exactly what adversaries are anticipated. In this section we will describe a hypothetical adversary (hereafter 'the adversary') against whom we will protect our users.

The adversary will be granted all powers available to all conceivable attackers, such that no collusion of attackers may overcome our security (should it work for any given considered attacker).

The following individual attackers are considered, those attackers excluded are excluded on the basis that their abilities are a subset of the union of the abilities of the already considered attackers.

- Nation state without regard for international law and convention (e.g.: USA)
  - Pressure those it claims governance over into doing as it demands
  - Pressure companies operating within it into colluding in an attack
  - Identify all people connecting to the server. (Formed from the union of powers of the ISP and the server owner and operators)
- ISP (e.g.: BSkyB)
  - View all traffic on their network, after the point at which a user comes under suspicion.
  - Manipulate all traffic on their network however they desire.
  - Identify an IP address (during a specific time) with a person.

- Server Owners and Operators (i.e.: Those who own and operate Turtlenet)
  - Alter the source of the server in any way they desire.
  - Log all traffic before and after a user comes under attack.
  - Manipulate all traffic in any way they desire.
  - Collect the IP of all connecting users.

Given that our system is intended to both protect people from the governments which claim governance over them, and mere greedy companies looking to sell or collect user data for profit, we will assume the worst case: i.e. that all our users, their ISPs, and the owners and operators of the Turtlenet server they use are able to be pressured by the adversary.

We grant the adversary all the powers listed above, and assume that all ISPs, companies, and Turtlenet server operators are activly working against all of our users. In summary, we consider the adversary to be:

> A nation state for which money is no object, claims governance over the user, and has the ability to pressure service providers into spying on their users.

4.1 Scope

We do not attempt to protect against an adversary who has access to and the ability to modify the users hardware, nor do we attempt to conceal that an IP uploads data to the network.

5

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.

6

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 architecture, for future maintenance
- Full description of database design
- Interface documentation

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

7

Anticipated Experiments and their Evaluation

7.1 Performance Testing

Evaluating how well the system performs under a high work load.

- Test to see how many simultaneous clients the server can handle.
- Test to see if the data received from the server under a high work load is accurate.
- Test the impact of a large number of clients on the servers response time.

A high work load will be simulated by automated clients performing user actions at random. The server should be capable of allowing each of these clients to communicate with another quickly.

7.2 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.

7.3 Recoverability Testing

Evaluating how well the application recovers from crashes and errors.

- Restart the computer while the application is running. Ensure the local database is not corrupted.
- While the application is running terminate the computers network connection. Ensure the application continues working after the connection is re-established.
- Send a badly formatted message to another client. Ensure the application is able to keep running after receiving unexpected data from another client.

7.4 Learnability Testing

Trialling the user interface with non expert users. Users should be able to use the system with minimal frustration and, ideally, without consulting the manual.

- Ensure users understand how to add friends, send messages, create posts, comment on posts and like posts.
- Ensure users don't spend excessive time searching for functions within the interface.
- Ensure error messages can be understood by the user and offer understandable advice on how to proceed.

7.5 Security Testing

The main goal of the system is to be secure. To ensure this goal is met the security of the system should be tested.

- Send non standard messages to clients. These should be rejected. If there is a flaw in the system the client may reveal information unintended for the recipient, in this case the program sending non standard messages.

- Recruit experienced programmers from outside of the group to attempt to penetrate or otherwise break the system. All attempts should be unsuccessful.
- Simulate a denial of service attack. Evaluate how well the system deals with the attack.

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 publicly 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

In order to register one must read and agree to their terms [6], read their data use policy [5], and read their cookie policy [4]. 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.[6, p. 4.1]

8.3 Account Management

The user is given the ability to set the security defaults for their posts and information. These 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

Access may be gained to an account by knowing certain information, the intent is to allow people to recover their account if they forget their password.

A users profile may contain the following information:

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

Field	Description
Photo	All the photos the user's has tagged
Friend	What friends the user has
Note	What notes the users up/downloaded to facebook
Groups	What groups the user has join
Events	What events user may be attending
Likes	What page(s) (unknown type) the user liked
Apps	What apps the user has
Books	What book pages the user liked/followed
TV programmes	What TV pages the user liked/followed
Films	What films pages the user liked/followed
Music	What music(or stars) the user liked/followed
Sports	What sport pages the user liked
Place	Where's the user has been

Table 8.1: The 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 another's posts

A post will typically be displayed on the news feeds 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 another's posts all together, merely prevent them from being automatically displayed on their news feed. 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'?

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

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.

8.6 wall

A users wall stores all the posts of the user posted since the account was created and the information about the user, this information is presented in reverse chronological 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 owner 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 Architecture

From a users point of view facebook is ostensibly organised as a single central server; we are here concerned with the general architecture and not the specific implementation of it, and so we will consider all of facebooks servers to be a single server for the purposes of this section.

Users connect to facebook using a web browser, and proceed to download a client written in javascript. User data is uploaded to facebook over HTTP as cleartext. The data is stored on unencrypted on facebooks servers, and facebook maintains a database of all data.

This allows clients to download only the data they need, as they can simply ask for it. This in turn means that facebooks current architecture can, and does, support a huge user-base, measured in the millions.

8.9 Security

In order to use facebook after registration a user must 'log in'. This places an authentication cookie on the users computer which gives anyone in possession 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

If the user logs in from an IP associated with a region geographically far from the last login, facebook will confirm that the user owns the account by asking them to identify a friend in a photograph, or by other means.

Facebook chat turns the users computer into a server, whereby facebooks central server sends messages to the client as it receives them, rather than the client requesting new messages. This has been used in the past to identify facebook users by correlating sent messages of specific size sent at a specific time, with , and collect evidence toward prosecuting people.

citation needed

Facebook has access to all its users data, and is able to erase, modify, and fabricate it. Facebook is aware of everything which happens on facebook. Censorship is a common occurrence on facebook.

9

Case Study: Tor

9.1 Overview of Protocol

Tor is an implementation of onion routing, it routes traffic from your computer through a number of other nodes; the final 'exit' node the routes the traffic to the final destination. Node IP's are listed publically in directory servers. In this manner to IP of clients connecting to a server is obscured from that server.

RSA/AES is used to ensure that only you, the exit node, and the final destination see the plaintext traffic being routed. With the use of TLS, SSL, or other end-to-end encryption those who see the plaintext can be reduced to you and the final destination. However a malicious exit node can MitM SSL connections using ssl-strip or a similar tool. There are methods of avoiding this, but it is a serious issue because users believe that SSL is secure. This exploit is found in the wild[10] and so is most definetly a concern.

Tor also supports 'hidden services' which seek to conceal the IP of the client from the server, and the IP of the server from the client. These are significantly more secure as the traffic never exits the tor network, however provide no protection from the adversary as will be described later; after all, we're assuming the server operators are colluding, so they will provide data required for traffic confirmation.

9.2 Security

Given that Tor is a low-latency network traffic can easily be correlated. This problem is ameliorated in high-latency netowrks such as mix nets, but not eliminated.

Tor does not seek to protect against size correlation or time correlation of traffic. Rather the purview of tor is to conceal the IP address of a client from the servers which it connects to.

Should a global passive adversary have perfect visibility of the internet, they would be able to track tor traffic from source to host by correlating the size and time of transmissions.

> The Tor design doesn't try to protect against an attacker who can see or measure both traffic going into the Tor network and also traffic coming out of the Tor network[3]. - Roger Dingledine

While such an attack amy initially sound infeasable, access is only needed to the client, the host, the entry guard, the exit node, and the internal node. So the question becomes: how likely is it that an adversary will have access to these five nodes?

We can safely assume that the adversary has access to the clients traffic, since our threat model is that of a nation state seeking to spy on its citizens. Furthermore we may assume that the adversary has access to the content host, as our threat model assumes that service operators may be pressured legally or otherwise into spying on their users. Therefore we must conclude, at least for *the* adversary, that Tor is unsuitable for concealing activity in traditional social networks.

Does this then mean that Tor is insecure? No. So far as we know[18] the US does not currently have the ability to reliably and consistently track tor users; if the US is incapable of doing so, it is reasonable to assume that no other nation state has this ability. This is however not something which should be relied upon, therefore we shall consider Tor as unsuitable for transmitting our data, at least if we were to do so as a traditional social network.

> With manual analysis we can de-anonymize a very small fraction of Tor users, however, no success de-anonymizing a user in response to a TOPI request/ on demand[18].

10

Case Study: GPG and Email

GPG is an implementation of the PGP[1], providing both public/private key encryption and also a number of symmetric ciphers that can be used seperatly.

It is common practice to use GPG to encrypt email, and several popular addons for browsers exist to aid in this[**gpgaddon**]. Unfortunetly gpg itself is difficult to use[11], and a significant barrier to entry.

The encrypting of email with RSA[1] is a good solution if one wants to keep the content of messages secure, and unmodified. However it is out of scope for PGP to hide whom is communicating, so while we find the underlying cryptography sound, our scope is simply too different for PGP to be of any use; with one exception.

Public key distribution is a significant challenge [2]. PGP partially solves this problem by introducing the concept of a 'web of trust'. In such a system one marks public keys as trusted, presumably the keys of people you trust, and the people whom you have marked as trusted can then sign the keys of other people whom they trust. These keys may then be distributed, with the RSA signatures of everyone who signed them, to everyone. If I download a key and see that it has been verified by someone that I trust, then I can trust that key (albeit less than the original key). This in combination with the small word

[1] and a symmetric cipher

[2] Our system can't do it, or it would be trivial to MitM users who don't check they recived the correct key via another channel

hypothesis[3][20] allows a large number of public keys to become known to a user merely by adding one friends key, and having the client automatically sign all keys it comes across from a trusted source.

We will take the 'web of trust' into consideration during design.

[3]The phenomenon that people in earths population seem to be seperated by at most 6 intermediaries.

11

Case Study: alt.anonymous.messages and Mix Networks

alt.anonymous.messages is a newsgroup to which people publically post encrypted messages. In order to retreieve messages a recipient downloads all new messages and attempts to decrypt them all, those which they are able to decrypt are read, and others ignored.

This type of system is known as a 'shared mailbox', and is often not used by hand, but by mix network servers, which provide high-latency email forwarding, and handle the encryption on behalf of the users. Mix-networks massivly slow timing-based traffic confirmation because they cache a large number of messages before sending them all out at once in a random order.

This system provides the property we are seeking: concealing who talks with whom on our network, even from the server itself. This property is ensured by the fact that the server cannot tell who reads a specific message, even though it knows which IP uploaded it. It also introduces a huge amount of overhead, in the form of downloading everyone elses messages as well as ones own.

Mix networks however have some serious issues, and misconfiguration easily allows for traffic correlation[16], albeit not confirmation (without a large sample size). Furthermore mix networks only function if the operator is trusted, this is unacceptable vs the adversary. For these reasons we will not use the idea of mix networks.

We have identified the method of operation of shared mailboxes as the basis for our communications protocol, and will build a social netowrk atop this concept.

12

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 to 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 another's public key, they will first be informed of that persons username, and prompted to categorise the person. The user will be able to create a number of categories 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, co-worker); a number of

individuals; or any combination thereof.

Upon receiving 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.

13

User Requirements

consider implementing a WOT system with levels of trust

13.1 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 into the client. The database will be encrypted using the same AES key as the keys are encrypted with.

13.2 Interacting with other users

People are add 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, depending 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 categories such as friends, family, and co-workers. 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)

13.3 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 updated by the user by future PDATA messages.

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

13.4 Account recovery

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

13.5 Posts

13.5.1 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 themselves 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 publicly. Users may also post to another users wall.

Wall posts may contain links to other content, however this content is never thumb-nailed[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.

13.5.2 Commenting and Liking

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.

Any wall post, comment, or other item on a wall may be liked.

13.5.3 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 otherwise normal wall posts.

The user has the option of setting a category of people as people for whom they desire to be alerted of events regarding.

13.6 Chat

Users may chat in real time, however messages can still be sent when one user logs off, to be received when they log in. Past conversations are saved, and a

[2]client MUST NEVER thumbnail links or otherwise access external content without EXPLICIT user consent (see tor/js exploit on freedom hosting by the USA and tracking techniques recently thwarted by GMail now caching images. Specifically the fact that by delivering content over a secure channel that initiates communications outside of that channel, the recipients of content may be identified. A common variation of this is 'pixels' whereby a would be tracker embeds a 1x1 transparent png in a document, and watches who downloads that image from their servers.[19]

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

14

System Requirements

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 throughout:

- A users average message posted to their wall is 200 characters
- A users average number of messages posted to their wall per day is 10
- A users average number of friends is 100 (each and every friend represents one key exchange)
- A users average private message (to single user) is 50 characters
- A users average 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 therefore 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 network 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 14.1: Hours a computer is turned off per day vs minutes to sync

We feel that waiting 2-5 minutes 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 inherently limited network size (<1500 users of one server is practical) we recommend 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 receiving messages, and also who those messages are sent to/from (although they couldn't know what the messages said).

The server therefore merely needs a fast internet connection to upload and download content from clients. The client is required to perform a significant amount of encryption and decryption, however the client will almost certainly be able to encrypt/decrypt faster than a connection to the internet so the network speed may be considered the limiting factor for users on the internet. Large companies however may very well use the system over a LAN, however these can be reasonably expected to have fairly modern computers which can more than handle RSA decryption.

15

Required Data

...

16

Transaction Requirements

There are 3 categories of data transaction:

- 1. Data entry
- 2. Data update and deletion
- 3. Date queries

16.0.1 Profile creation of the user

The user is required to own an account with Turtlenet in order to log in. If no account has been made, the user is required to create one before proceeding to use the system. In order to create one, the user has to enter his profile information such as his name, date of birth, gender, and other necessary attributes. Once user has submitted it, it will be inserted into the database.

16.0.2 Adding of user relations

The term relations for this case means that the user has added has added the other particular user into the list of the people whom he knows. Note that the word 'friend' isn't used here mainly because a related user can fall into different sorts of categories which is not necessarily called 'friends'.

When a user adds another user as a relation, a public key of the other user has to be acquired first before the relationship between the two takes place. When the public key is acquired and inserted into a field, the database will use the key as a reference to get the details of the user. Once it is recognised,

the system then displays the user information whilst adding him into a list of relations.

16.0.3 Assigning relations into categories

When a user adds a relation, he has a choice of adding him onto a specific category (or categories). A user can create any category he wants by going to the options and click 'Add new category'. The database then records the new category into the category table. The user then can then assign the relation into the existing category.

16.0.4 Adding of posts

When a user adds a new post, details such as creator of the post, content, date and time of when the post has been created is inserted into the database. The user has a choice of making his post either public, shared amongst a certain category or shared with only a group of specific users. The user has the click the names of those who can view the posts, and these names will be inserted into the post table in the database.

16.0.5 Adding of events

This works similarly like adding a new post. However the only difference is details such as date of the event is inserted into the database when a new event is created. The security function is the same with post where the user can add others to view their event. The details of these users are inserted into the events table.

16.0.6 User creating a new message

A user can initiate a conversation with another user by creating a new message. When message is created, the public key is generated for user's relation to obtain for decrypting the message. The message will then be logged by inserting the time and date into the message table by the time the message is sent. Other details such as the sender's and receiver's user_id, content of the message will be inserted into the message table as well.

16.0.7 Receiving and unlocking a message

When the user receives a message from a relation, a notification will be sent to he user indicating that there is a new message. Simultaneously, this will record the encrypted message into the message table, along with the logged details, which is the time and date of the message. When the user enters the correct public key, the message will the by decrypted, and this will update the message table with the new contents of a decrypted message.

17

Risk Assessment

17.1 Parallel Tasks

A big concern for any project is the amount of tasks that will be performed simultaneously [2]. 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.

17.2 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.

17.3 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.

17.4 Scope

The scope is what the project will be encompassing and therefore is one of the most important sections as it defines what you'll be doing for the entirety of the project. That's not the only risk associated with the scope [12]. There is also scope creep, which is when the scope grows to cover more work than the project originally intended, often without an increase in resources matching the higher load on the project deliverables. Performing estimates on the scope, as well as anything else in the project, can be inaccurate as you are essentially guessing the near future which is difficult at the best of times.

To minimise the risk placed upon the scope, it is best to define what exactly is required of the project before any work takes place on the deliverables. For example it is best to define an encryption method for a project at the beginning and sticking to it rather than changing the method which may require a different implementation, creating more work. If at a later phase ambiguities appear in the scope, a meeting to define or even redefine these points should occur before any more work is carried out on the offending article, reducing the amount of change to the project that shall occur.

17.5 Change Management

Change Management is the application of a structured process and set of tools for leading the people side of change to achieve a desired outcome [9]. Problems that are associated with Change Management include conflicts which occur between stakeholders, as they may be disagreeing in how the project should move forward, an assumption that an irreparable state has befell the project due to a drastic amount of changes that have been placed upon the project or even ambiguous or inaccurate changes being added onto the project [12]. All of these can amount into an increase in workload or a decrease if the targets haven't been defined properly.

To reduce the amount of risk involved with Change Management, communication and clear definition on what the project needs to perform is required. Stakeholders should be as detailed as possible at every stage so that no ambiguity is caused, or cleared up if any does occur.

17.6 Stakeholders

Stakeholders are people that have an interest in the project, whether they are the members of the group, the group's monitor/superior or the target audience of the project. Some of the problems that Stakeholders cause for the project members include losing interest - if they become uninterested with the project then they may back out, which can be dangerous for the project if they were providing any form of input, such as experience in the target field or economic support. Most Stakeholders are disillusioned when it comes to the project - they are unaware of what the deliverables will be or have a twisted view on what and how the final product will perform its intended purpose. As always there is also a risk in terms of quality - Stakeholders may give ambiguous input both accidentally or on purpose, depending whether the Stakeholder wants the project to fail or not [12].

The best way to reduce the amount of risk involved with Stakeholders would be to keep them informed of the project's current status through external communication such as e-mail and through meetings so that the team can personally inform the Stakeholder with relevant information which should ease their mind of any apprehensive thoughts about the project [7].

17.7 Platforms

The main risk in Platforms would be the difference between the chosen development platform and the target market's system. The change between executable files for different operating systems are usually great enough so that a separate executable is required for each distinct operating system. What may also cause problems, especially with low-level programming, would be differing architectures, hardware sets and how the system reads commands [15]. Another problem with platforms would be whether the required software for the project is installed, such as any required run-time environments or files which are needed to use Structured Query Language databases.

This risk can be eliminated if platform-independent code is used - such as the Java Programming Language [8]. This would mean that no changes in implementation would be needed and database functionality could occur within the platform-independent environment if need be. Otherwise to reduce the amount of risk involved with the varying systems that the target audience may own, compiling the source on different virtual systems to create executables for the

many various platforms available would suffice. Of course, this can be mitigated by choosing to not support other systems in favour of only allowing the development platform and its Operating System to be supported.

17.8 Integration

The integration of the project can be high risk due to a couple of factors:

- The intended environment is incompatible or unavailable
- Incomplete testing means the final product may be buggy
- Final product doesn't work (e.g. bad link to database)
- Product lowers efficiency due to learning curve [12]

In order to combat the risks involved in implementation, having a set testing day in an isolated environment can allow the completed builds of the project to be evaluated before being given to the target audience. This will allow the checking of compatibility with the system as well as in-house bug testing. A manual or help section could be implemented into the system so that the learning curve is not as steep compared to not having such resources.

17.9 Requirements

Requirements are not just a list of functional needs and wants but also the constraints on the project as well. However, their are similar risks involved in the requirements, such as generalisation, ambiguity or even being incomplete. Another risk to do with requirements is whether they align with the design factor or not.

An example would be having both 'fast processing' and 'system independence' as requirements - C++ is faster but Java is independent of platform and although speed may not be an issue with smaller data, larger chunks of data will undoubtedly have an effect on interpreted code [14].

To minimise the risk with requirements, communication between group members and stakeholders is needed - making sure that the requirements and the scope are in line with each other and that any suggested changes are properly handled with little to no ambiguity. Choosing a design structure and sticking to it is also beneficial to the project. Reducing the workload of the implementation

can help towards minimising the risks of requirements and the program, such as removing old data that is no longer needed upon the program's start-up.

17.10 Authority

Without distinct authority within the project, risks can become apparent. If the members of the project do not have the correct privileges on the target system to perform what is required, work output slows or even stops until the matter is resolved. Another risk would be misguided authority - where the team is unclear who has been given the authority to perform a task and therefore there are multiple members allocating the same task to themselves, which will slow down the efficiency of the team due to duplicated work.

Lowering the negative impact of Authority is done through the use of clear definitions. Allocating work to project members and centralising a form of 'to-do' list so that project members can look up what has been assigned to them. Another way of reducing the amount of inefficiency caused by problems with authority would be to make sure the permissions are correctly set up on both the testing and target systems.

17.11 External

There are a couple of external factors which may impact the project in a negative manner. The first being any legal restrictions. This is important as there is a chance that the final product may be used in a location that differs to the geographical area that it was developed in. For example there is a law within the UK called the 'Key Disclosure Law' which means that you must give decryption keys to UK authorities [13]. In the United States however, it is something of a grey-area, as giving up encryption keys could violate the fifth amendment, as doing so could give incriminating evidence against yourself:

> 'unlike surrendering a key, disclosing a password reveals the contents of ones mind and is therefore testimonial.' [17]

Not only is the law a big risk in projects, but also nature. If you are situated where natural disasters can happen or otherwise things such as heavy weather occur, this can reduce the work flow by denying the team members access to their workspace. Another factor that is external is the changing of technology.

Updates to programming languages can lead to deprecated functions or newer operating systems may not be capable of running the same software as their previous iterations, meaning an increased amount of work to keep the software compatible with the target system.

Reducing the amount of risk caused by external factors is difficult as the project team have little to no influence upon them. For example the team cannot bypass any laws that govern the area that the program will be used in so they must be adhered to as part of the constraints of the project. Natural disaster cannot be stopped but if you are able to, bringing some of the work back so you could work on it during bad weather may reduce the impact that said weather will have on the project. To reduce the damage caused by software deprecation it is ideal if the functionality coded in the project is not old, or otherwise buggy, so that maintaining or updating the software will require less work.

17.12 Project Management

Project Management, or rather a lack of, can also be a risk to the endeavours of the team. If the group has been asked to reduce or combine the amount of stages in the System Development Life Cycle (SDLC), this can increase the risk of the project failing because it leaves more room for error - combining the stages will often cause a decrease in quality as less resources are being dedicated to a particular section of the project. A lack of Project Management will also be seen as a high risk because of how difficult it is to monitor a project and its success without these tools.

To reduce the risk that Project Management will apply upon the project, a formal methodology, such as the 'waterfall' method could be implemented. This would however reduce inefficiency as the output needs to be moderated and cleared before the start of the next stage in the SDLC can occur. On the other hand an informal methodology would increase the risks but may potentially allow the project to be completed within a smaller time frame and to the same standard.

17.13 User Acceptance

Just because a project has been made for a target audience doesn't mean that *that* audience will like it. During testing the target market may reject the initial builds of the project due to the way it does or does not work or the look of the project could mean that it is unwieldy to use, whether it is due to low quality or the interface being anti-intuitive.

The main method of reducing the risk pre-emptively is to perform research on any currently available software that achieve similar goals to the project's. By doing this you can find out what users are acquainted with and create a similar yet unique design or use the competitors as a way of highlighting what is wrong with the current market and create something entirely different. Another method which does require more work is to take in user feedback during testing and implement their suggestions for the look of the project, or the inner mechanics if they have the knowledge to suggest improvements.

17.14 Conclusion

In order to reduce the risk of the project as a generalisation, it is suggested that you:

- Have a centralised communication system used by all members - this reduces all communicative related risks.
- Define team objectives and allocation clearly - this reduces the authority-based risks as well as any that are communicative.
- Define a target system for development - other types of platform can be supported at a later date should the need arise.
- Create and uphold a work ethic to be followed by everyone - this helps to maintain a standard of quality throughout the project.
- Testing should be first on each individual module/deliverable, then as a whole. This improves bug catching and helps monitor the quality of the project.
- Choose a methodology and follow it - this creates a standard of work ethics which will give a layout as well as structure to the project.

By following these pointers a moderate amount of risk can be mitigated with little need for concern. Do note that the legality of the project in differing countries should be researched and followed, should the project be in use within that country.

18

Implementation Stage and Planning

18.1 Task List

Task ID	Task Description (Desc.)	Due Date	Deliverable
1	Project Planning	14/02/2014	Planning segment
1.1	Mission Statement	07/02/2014	Same as Desc.
1.2	Mission Objectives	07/02/2014	Project Goals
1.3	Project Target	07/02/2014	Project Scope
1.4	Requirements	07/02/2014	Project Scope
1.5	System Requirements	07/02/2014	Same as Desc.
1.6	User Requirements	07/02/2014	Same as Desc.
1.7	Transaction Requirements	07/02/2014	Same as Desc.
1.8	Case Studies (CS)	14/07/2014	Eval. of rival
1.8.1	CS: Facebook	14/07/2014	Eval. of rival
1.8.2	CS: 'GPG' and E-Mail	14/02/2014	Eval. of rival
1.8.3	CS: 'Tor'	14/02/2014	Eval. of rival
1.9	Required Data	14/02/2014	Same as Desc.
1.10	Risk Assessment	14/02/2014	Same as Desc.
1.11	Anticipated Software	14/02/2014	Project Estimates
1.12	Anticipated Experiments	14/02/2014	Project Estimates
1.13	Anticipated Documentation	14/02/2014	Project Estimates
1.14	Methods of Evaluation	14/02/2014	Same as Desc.
1.15	User View	14/02/2014	Same as Desc.
1.16	Gantt Chart	14/02/2014	Same as Desc.

Task ID	Task Description (Desc.)	Due Date	Deliverable
2	Project Design	14/03/2014	Design Segment
2.1	Research (Res.)	21/02/2014	Research Segment
2.1.1	Res: Database Languages	21/02/2014	Same as Desc.
2.1.2	Res: Programming Languages	21/02/2014	Same as Desc.
2.1.3	Res: Interfaces	21/02/2014	Same as Desc.
2.2	Designs (Des.)	07/03/2014	Design Segment
2.2.1	Des: Databases	28/02/2014	Same as Desc.
2.2.2	Des: Class Interfaces	28/02/2014	Same as Desc.
2.2.3	Des: Protocol	28/02/2014	Same as Desc.
2.2.4	Des: Architecture	28/02/2014	Same as Desc.
2.2.5	Des: Sequence Diagrams	28/02/2014	Same as Desc.
2.2.6	Des: Data Flow Diagrams	28/02/2014	Same as Desc.
2.2.7	Des: Class Diagrams	28/02/2014	Same as Desc.
2.2.8	Des: Server-side Interfaces	28/02/2014	Same as Desc.
2.2.9	Des: Client-side Interfaces	28/02/2014	Same as Desc.
2.2.10	Des: Server-side Protocols	28/02/2014	Same as Desc.
2.2.11	Des: Client-side Protocols	28/02/2014	Same as Desc.
2.2.12	Des: Server-side Pseudo-code	07/03/2014	Same as Desc.
2.2.13	Des: Client-side Pseudo-code	07/03/2014	Same as Desc.
2.3	Segment Review	10/03/2014	Design Segment
2.3.1	Evaluate Segment Quality	14/03/2014	N/A
2.3.2	Improve Segment	14/03/2014	Design Segment

Task ID	Task Description (Desc.)	Due Date	Deliverable
3	Implementation stage (Imp.)	28/04/2014	Imp. Segment
3.1.1	Imp: Architecture	21/03/2014	Work Environment
3.1.2	Imp: Architecture Docs	21/03/2014	Documentation
3.2.1	Imp: Target System (TS)	21/03/2014	Work Environment
3.2.2	Imp: TS Documentation	21/03/2014	Documentation
3.3.1	Imp: Databases	21/03/2014	Database
3.3.2	Imp: Database Documentation	21/03/2014	Documentation
3.4.1	Imp: Server-side Protocols	28/03/2014	Program function
3.4.2	Imp: Server Protocol Docs	28/03/2014	Documentation
3.5.1	Imp: Client-side Protocols	28/03/2014	Program function
3.5.2	Imp: Client Protocol Docs	28/03/2014	Documentation
3.6.1	Imp: Server-side Interface	04/04/2014	Interface
3.6.2	Imp: Server Interface Docs	04/04/2014	Documentation
3.7.1	Imp: Client-side Interface	04/04/2014	Interface
3.7.2	Imp: Client Interface Docs	04/04/2014	Documentation
3.8.1	Imp: Server-side Source Code	18/04/2014	Program
3.8.2	Imp: Client-side Source Code	18/04/2014	Program
3.9.1	Imp: Server Install Docs	18/04/2014	Documentation
3.9.2	Imp: Client Install Docs	18/04/2014	Documentation
3.10	Segment Review	18/04/2014	Imp. Segment
2.10.1	Evaluate Segment Quality	28/04/2014	N/A
2.10.2	Improve Segment	28/04/2014	Imp. Segment

Task ID	Task Description (Desc.)	Due Date	Deliverable
4	Project Portfolio	09/05/2014	Portfolio
4.1	Fabricate Reports	02/05/2014	Reports

18.2 Gantt Chart

gantt chart, along with discussion of it's value and applicability

18.3 System Boundary Diagram

description of diagram, and why it is useful

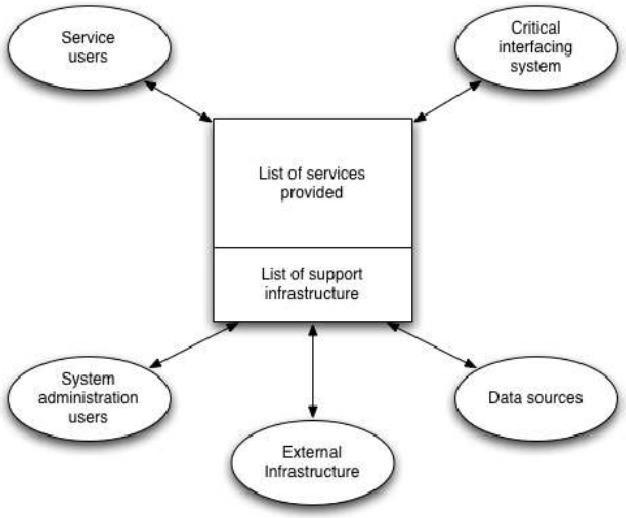

Figure 18.1: System Boundary Diagram

Part II

Design

19

Protocol

19.1 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.

20

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 20.1: table: users

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

Table 20.2: 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 20.3: table: wall_post

attribute	description
id **PK**	
login_time	
logout_time	

Table 20.4: table: login_logout_log

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

Table 20.5: table: private_message

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

Table 20.6: table: comment

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

Table 20.7: table: like

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

Table 20.8: table: events

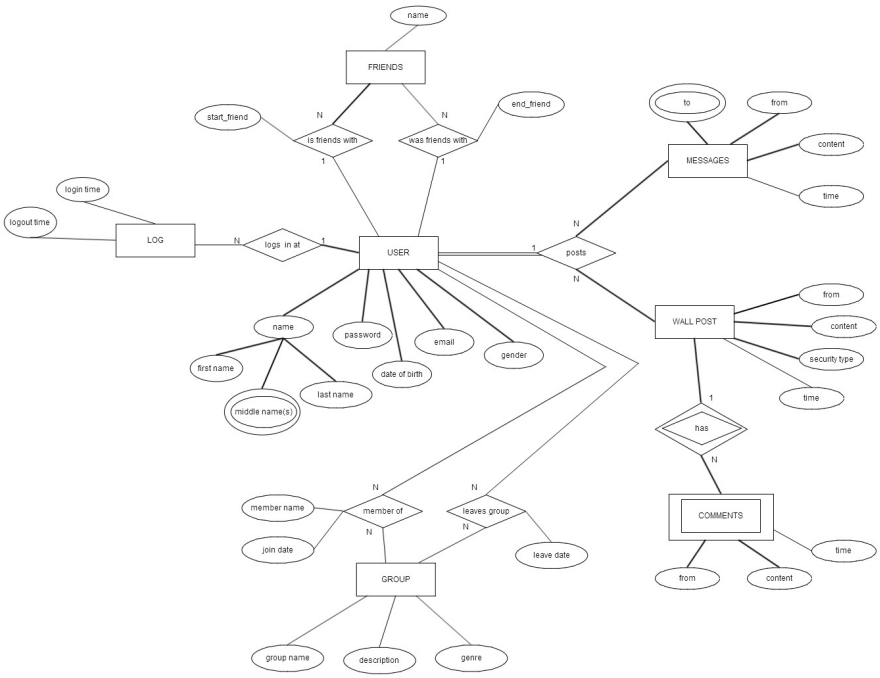

Figure 20.1: Database E-R Diagram

21

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

21.1 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 21.1: 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 21.2: 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 person(s)
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 21.3: 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 21.4: user adds a new event

field	notes
comment_id	
post_id	
comment_from	
comment_time_ date	

Table 21.5: user adds a comment

field	notes
like_id	
post_id	
like_from	

Table 21.6: 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

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.

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

2. Statement of Deliverables

- **DRAFTED** Desc. of anticipated documentation (Luke)
- **DRAFTED** Desc. of anticipated software (Aishah)
- **DRAFTED** Desc. + Eval. of any anticipated experiments + blackbox (Louis)

- **INCOMPLETE** System boundary diagram (Leon)
- **DRAFTED** User view and requirements (Luke)
- **DRAFTED** System requirements (Luke)
- **DRAFTED** Transaction requirements (Aishah)

3. Project and Plan

- **DRAFTED** Facebook research (Leon)
- **DRAFTED** Case Study: Tor (Luke)
- **DRAFTED** Case Study: alt.anonymous.messages and mix networks (Luke)
- **DRAFTED** Case Study: PGP and E-Mail (Luke)
- **IRRELEVENT?** Data required (???)
- **INCOMPLETE** Implementation Stage (???)
- **DRAFTED** Milestone Identification (Milestones can most easily be recognised as deliverables) (Mike)
- **INCOMPLETE** Gantt Chart (Mike)
- **INCOMPLETE** Pert Chart (Mike)
- **DRAFTED** Risk Assessment (Mike)

4. Bibliography

- **COMPLETE** Bibliography framework (Luke)
- **INCOMPLETE** Add citations where relevent (Everyone, in their own sections)

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] J. Callas et al. OpenPGP Message Format. English. RFC. Nov. 2007. URL: https://tools.ietf.org/html/rfc4880 (visited on 02/11/2014).

[2] Oracle Corporation. The Benefits of Risk Assessment on Projects, Portfolios, and Businesses. English. White Paper. Enterprise Software, Computer Software, June 2009. 14 pp. URL: http://www.oracle.com/us/042743.pdf (visited on 02/10/2014).

[3] Roger Dingledine. One cell is enough to break Tor's anonymity. English. Tor Project. Feb. 2009. URL: https://blog.torproject.org/blog/one-cell-enough (visited on 02/11/2014).

[4] Facebook. Cookies, Pixels & Similar Technologies. English. 2014. URL: https://www.facebook.com/help/cookies (visited on 02/10/2014).

[5] Facebook. Facebook Datause Policy. English. Nov. 2013. URL: https://www.facebook.com/about/privacy (visited on 02/10/2014).

[6] Facebook. Statement of Rights and Responsibilities. English. Nov. 2013. URL: https://www.facebook.com/legal/terms (visited on 02/10/2014).

[7] Amanda Dcosta/Marlene Gundlach. Stakeholders Risk Management - Project Risk Assessment Ap
English. Bright Hub Inc. Feb. 2014. URL: http://www.brighthubpm.com/risk-management/33399-stakeholders-risk-management-project-risk-assessment-approach/ (visited on 02/10/2014).

[8] Oracle Inc. Java. Features and Benefits. English. Oracle Inc. Feb. 2014. URL: http://www.oracle.com/us/technologies/java/features/index.html (visited on 02/10/2014).

[9] Prosci Inc. Change Management: The Systems and Tools for Managing Change. Defining Change Management English. Prosci Inc. Feb. 2014. URL: http://www.change-management.com/tutorial-change-process-detailed.htm#Definition (visited on 02/10/2014).

[10] Known Bad Relays. English. Tor Project. Jan. 2014. URL: https://trac.torproject.org/projects/tor/wiki/doc/badRelays (visited on 02/11/2014).

[11] Peter Maass. How Laura Poitras Helped Snowden Spill His Secrets. English. Aug. 2013. URL: http://www.nytimes.com/2013/08/18/magazine/laura-poitras-snowden.html?smid=pl-share (visited on 02/10/2014).

[12] Anna Mar. 130 Project Risks (List). English. Mar. 2013. URL: http://management.simplicable.com/management/new/130-project-risks (visited on 02/10/2014).

[13] Pinsent Masons. Law requiring disclosure of decryption keys in force. English. Pinsent Masons LLP. Oct. 2007. URL: http://www.out-law.com/page-8515 (visited on 02/10/2014).

[14] J.P.Lewis/Ulrich Neumann. Performance of Java versus C++. English. Comparison. California, USA: Computer Graphics and Immersive Technology Lab - University of Southern California, 2004. URL: http://scribblethink.org/Computer/javaCbenchmark.html (visited on 02/10/2014).

[15] General Public. x86-64. Differences between AMD64 and Intel 64. English. Wikimedia Foundation Inc. Feb. 2014. URL: http://en.wikipedia.org/wiki/X86-64#Differences_between_AMD64_and_Intel_64 (visited on 02/10/2014).

[16] Tom Ritter. "De-Anonymizing Alt.Anonymous.Messages". In: Defcon 21, Nov. 2013. URL: https://www.youtube.com/watch?v=_Tj6c2Ikq_E (visited on 02/11/2014).

[17] Proskauer Rose. UK Court Parts with US Court regarding Compelled Disclosure of Encryption Keys. English. Proskauer Rose LLP. Oct. 2008. URL: http://privacylaw.proskauer.com/2008/10/articles/international/uk-court-parts-with-us-court-regarding-compelled-disclosure-of-encryption-keys/ (visited on 02/10/2014).

[18] <Top Secret>. Tor Stinks. English. NSA. June 2012. URL: http://media.encrypted.cc/files/nsa/tor-stinks.pdf (visited on 02/11/2014).

[19] Richard M. Smith. The Web Bug FAQ. English. Nov. 1999. URL: http://w2.eff.org/Privacy/Marketing/web_bug.html (visited on 02/10/2014).

[20] J. Travers and S. Milgram. "An Experimental Study of the Small World Problem". English. In: Sociometry 32.4 (Dec. 1969). URL: http://www.cis.upenn.edu/~mkearns/teaching/NetworkedLife/travers_milgram.pdf (visited on 02/11/2014).