COMP208 - Group Software Project
Ballmer Peak

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

March 7, 2014

Contents

Part I

Requirements

Chapter 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.

Chapter 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[24]. 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. It is important to hide the recipient of the message, because otherwise they may be unfairly targeted[11] if they use our services to communicate with the wrong people on a phone which is later identified, or they may merely be 'selected' for spying and harassment [20, p. 3].

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.

> "ItâĂŹs really annoying and complicated, the [email] encryption software" - Glenn Greenwald [20]

While there exist many tools for hiding what you are saying, relatively few seek to hide who talks with whom, and those which do often implement it merely as a proxy, or seek to provide convenience over security.

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 unjust, seeking to spy on potential users, 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 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

[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

- 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.

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

Chapter 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[32] and a majority of the middle east[19] 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 (e.g. IBM, Thales, 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 project would match their needs well.

A more likely recipient of this system however, is society itself, as we have decided to waive our copyright granted monopoly. 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. See Appendix B.

Chapter 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.

Some of these claims may seem extreme, but given that companies such as BT, Vodafone Cable, Verizon, Level 3, and others have provided unlimited access to their networks[16] to governmental spy agencies, we feel it is a reasonable threat model in light of recent revelations[31].

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 actively 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.

While we recognise that the ability to post messages anonymously is important, especially considering that countries normally considered benign are prosecuting people over whom they claim governance for saying 'offensive' things [1], it is unfortunately out of scope for this project.

Chapter 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

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.

Chapter 6

Anticipated Documentation

We will provide the following documentation:

- Installation guide for a server
- User manual for a 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

Chapter 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 these clients to communicate with one another quickly. The maximum number of concurrent clients possible without noticeable lag (twice the frequency of updates) should be recorded.

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.

- Simulate a denial of service attack. The server should be able to recover from the attack quickly and with minimal impact on the clients. Blocking such an attack is beyond the scope of this project.

Inputs used should range from expected use patterns to silly as users tend to do things totally unexpected. Expected and actual outputs should be recorded. Any differences will indicate problems with the system which need to be fixed.

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.

Each test should be run several times. If any test fails once or more this indicates that the system is bad at recovering from crashes and/or failures. In the case of a failure changes to the system should be implemented to improve recoverability.

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.

Each test should be run several times with different users. If more than one user fails a test then changes need to be made to the interface. A single user experiencing problems is not an indication of a problem with the interface but instead suggests user incompetence.

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.

If any test fails this indicates a vulnerability in the system which should to be corrected immediately. Security tests should be rerun after any changes during the testing phase to ensure new vulnerabilities are not introduced.

Chapter 8

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, and a passphrase. 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. The entered passphrase will be used to deterministically derive an AES key which will be used to encrypt the users keypair and local DB. The user will be given the option of creating a second passphrase which, when entered, will overwrite the keypair and local DB with random data.

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 a QR encoded or plaintext encoded 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.

8.1 System Boundary Diagram

Each client (of which there may be many) has his own client boundary consisting of his database and program client, whilst the server operators have their own boundary consisting of just the server. You can see in the diagram that at no point does the server operator or user functionality coincide with each other, leaving their privacy fully independent of one another. Each client (of which there may be many) has his own client boundary consisting of his database and program client, whilst the server operators have their own boundary consisting of just the server.

We can see the users interaction with the system below in the System Boundary Diagram:

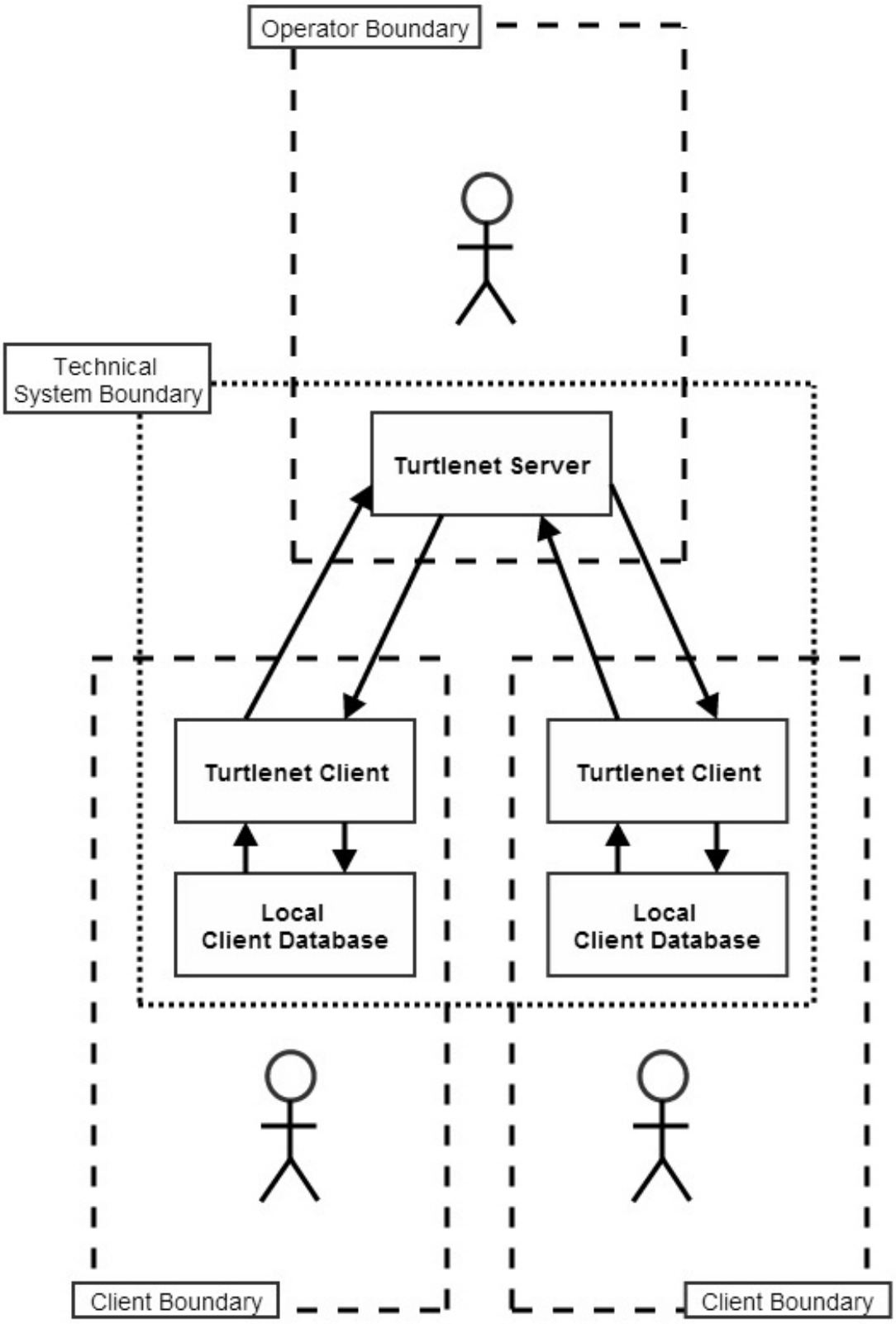


Figure 8.1: System Boundary Diagram

Chapter 9

User Requirements

9.1 Registration

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

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

9.2 Interacting with other users

People are added 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 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.

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

9.3 Profile Data

Profile data will be transmitted via PDATA messages. Different versions of profile information may be provided to different groups of people. Profile data may be 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
- Misc.

9.4 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.

9.5 Posts

9.5.1 Walls

Each user has their own wall. On their wall they may posts messages for themselves and others to see. All wall posts should be addressed to the user 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].

[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.[28]

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.

9.5.2 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 the 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.

9.5.3 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 a group for whom they desire to be alerted of events regarding.

Furthermore users may create their own events, for themselves and others to be alerted to. Recipients of events they did not create must accept the event before they are alerted of it.

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

Conversations may include two or more people.

Chapter 10

Case Study: Facebook

10.1 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.

10.2 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 [10], read their data use policy [9], and read their cookie policy [8]. 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.[10, p. 4.1]

10.3 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.

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
- Place of Birth
- Relationship
- Basic Information
  - Birthday
  - Relationship
  - Status
  - Anniversary
  - Languages
  - Religious
  - Political
  - Family
  - Contact Information

10.4 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

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 10.1: The user adds a new post

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.

10.5 Post

10.5.1 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

10.5.2 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 however, may be disabled by the user who makes the 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 (e.g. 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.

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.

10.6 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 along with it's 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.

10.7 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. You are also informed whether your friend is logged in on a mobile device or not.

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.

10.8 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.

10.9 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.

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.

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.

Chapter 11

Case Study: Tor

11.1 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[15]. Node IP's are listed publicly in directory servers. In this manner the 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[18], and so is most definitely 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.

11.2 Security

Given that Tor is a low-latency network, traffic can easily be correlated. This problem is ameliorated in high-latency networks 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[7]. - Roger Dingledine

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, due to traffic confirmation.

Does this then mean that Tor is insecure? No. So far as we know[30] 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, as assumptions widely lead to mistakes. We shall therefore 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[30].

Chapter 12

Case Study: GPG and Email

GPG is an implementation of the PGP[2], providing both public/private key encryption and also a number of symmetric ciphers that can be used separately.

It is common practice to use GPG to encrypt email, and several popular addons for browsers exist to aid in this[12]. Unfortunately GPG itself is difficult to use[20], 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 who 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 hypothesis[3][29] 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, however it may present some significant security issues.

[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 received the correct key via another channel

[3] The phenomenon that people in the earth's population seem to be separated by at most 6 intermediaries.

Chapter 13

Case Study: alt.anonymous.messages and Mix Networks

alt.anonymous.messages is a newsgroup[1] to which people publicly post encrypted messages. In order to retrieve 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 massively slow timing-based traffic confirmation because they cache a large number of messages before sending them all out at once in a random order[26].

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 else's messages as well as ones own.

Mix networks however have some serious issues, and misconfiguration easily allows for traffic correlation[25], albeit not confirmation (without a large sample size). Furthermore mix networks only function if the operator is trusted, this is unacceptable against our threat model. 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 network on top of this concept.

[1] It may be accessed, without an installed Usenet client, through several websites providing an interface to it, one such website is Google Groups, and may be accessed here: https://groups.google.com/forum/#!forum/alt.anonymous.messages

Chapter 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 [5]. 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.

Chapter 15

Transaction Requirements

Due to the nature of Turtlenet there may exist no central database, rather each client maintains their own local database of everything they know. The data forming this is all stored centrally, however to build a complete database would require the private key of every user of the service, which clearly we do not have access to.

There are 3 categories of data transaction:

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

15.1 Profile creation of the user

All that is required to use a Turtlenet server is a valid RSA keypair. Users don't have accounts per se, but rather associate profile data with a public key if they so desire. Users have no login information, rather posts are authenticated via RSA signatures. Usernames are the sole public information in our system, and as such each client has a complete list of usernames.

When a client first connects it is advisable, albeit not required, to claim a username. This is done merely by posting that username, and a signed hash of it to the server. Therefore the DB must store all such CLAIM messages.

Optional profile data which the user may enter is stored as PDATA messages, and the databse will be required to store these.

15.2 Adding of user relations

Communication between people on Turtlenet requires that one is in possesion of the public key of the recipient, and should they wish to respond then they must be in possesion of your public key. We define 'A being related to B' to mean that A is in possesion of B's public key, and B is in possesion of A's public key. This is given a special name as it is a very common situation.

A user may be uniquely identified by their public key, and it may be used to derive their username, if they claimed one. Being in a relation with someone doesn't mean that you can see any profile information of theirs, however the GUI will ask the user whether they wish to share their own profile information with someone when they add that persons key.

15.3 Assigning relations into categories

When a user adds a relation, he has a choice of adding him into 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.

Categories are useful because they allow the user to share their posts with a predetermined set of people automatically, withing having to list each individual as a recipient.

15.4 Adding of posts

Users may post on their, or - with permission - others, walls. A post has a list of people who can see it (the user may choose a previously defined category or a specific list of people) however this list isn't public so only the DB of the author of a post (and the owner of the wall) will contain information as to who is able to see it.

The post itself has a timestamp, a signature (authenticating it), and content. The databse will store all of these.

NB: When a user posts something, they are automatically added to the list of recipients. A users own posts are downloaded from the server, just like everyone elses, and are in no way special.

15.5 Adding of events

The database will store events, these may be created by the user, or recieved from other users. At the appropriate time the GUI will notify the user of an event occuring. Example include birthdays, deadlines, and important dates. Events recieved from other users must be accepted by the user

before the GUI will alert the user of them, for this reason the DB must also store whether an event is accepted or not.

15.6 User creating a new message

A user can initiate a conversation with (an)other user(s) by creating a new message. Messages are merely a special case of wall posts, which are handled differently by the GUI.

15.7 Receiving Content

When the client connects it will download all messages posted since it last connected, it will then attempt to decrypt them all using the users private key. Those messages which are successfully decrypted are authenticated by verfying the signature and the content added to the database. It is in this manner that all content is passed from server to client.

Chapter 16

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	Threat Model	07/02/2014	Project Scope
1.5	System Requirements	07/02/2014	Same as Desc.
1.6	User View and 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
1.8.4	CS: 'aam' and mix networks	14/02/2014	Eval. of rival
1.10	Risk Assessment	14/02/2014	Same as Desc.
1.11	Anticipated Software	14/02/2014	Project Estimates
1.12	Anticipated Experiments and Evaluation	14/02/2014	Project Estimates
1.13	Anticipated Documentation	14/02/2014	Project Estimates
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	28/04/2014	Imp. Segment
3.10.1	Evaluate Segment Quality	28/04/2014	N/A
3.10.2	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

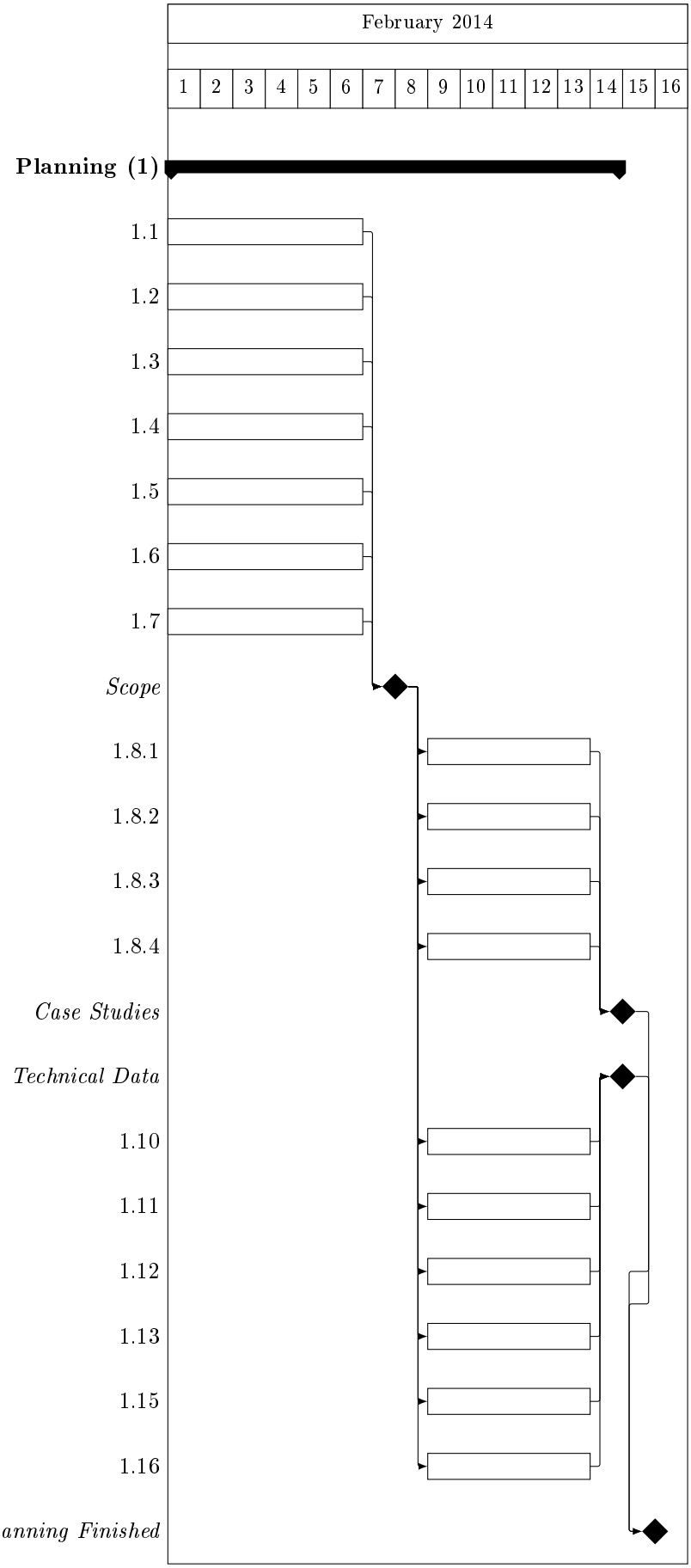
Chapter 17

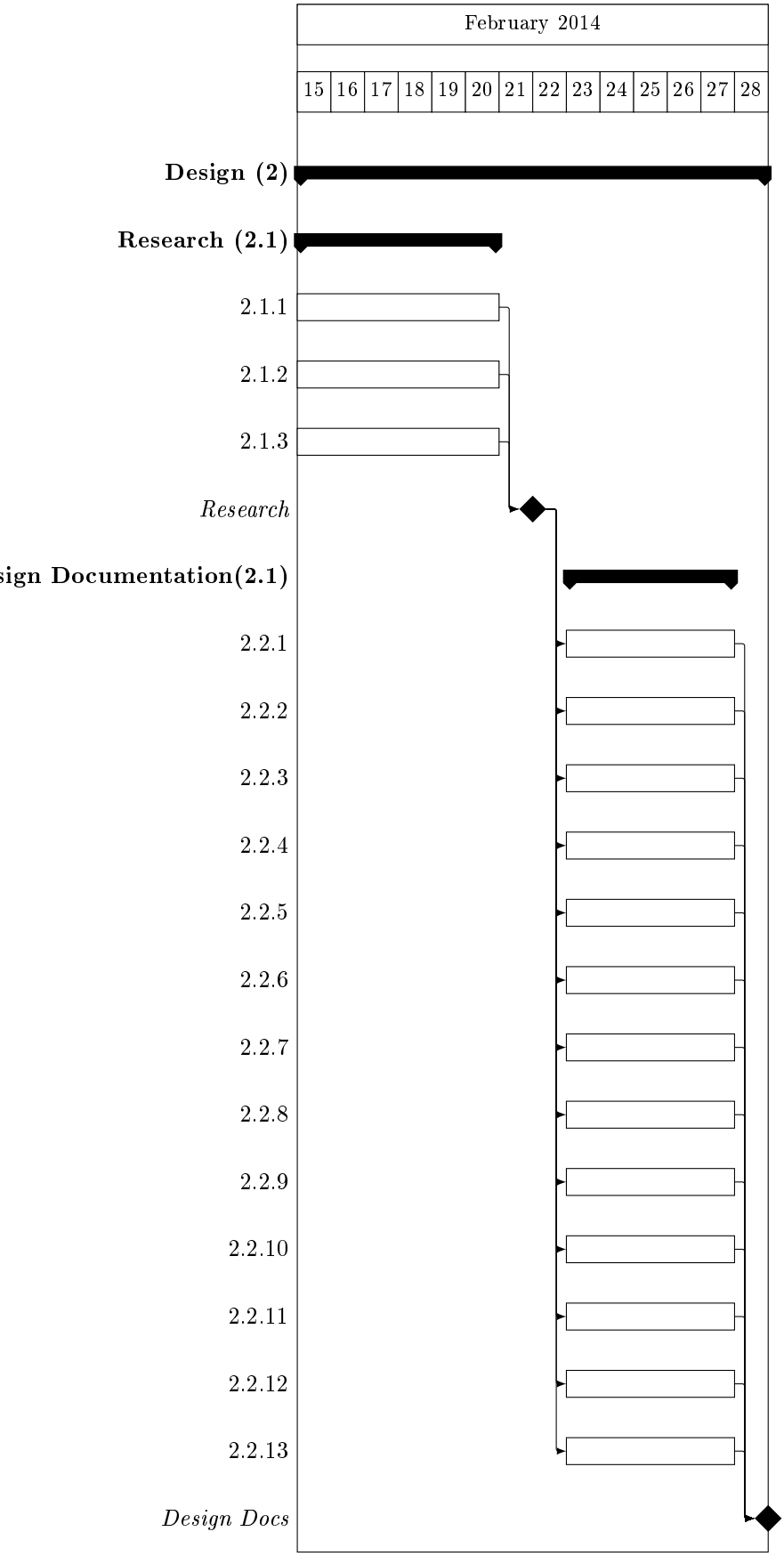
Gantt Chart

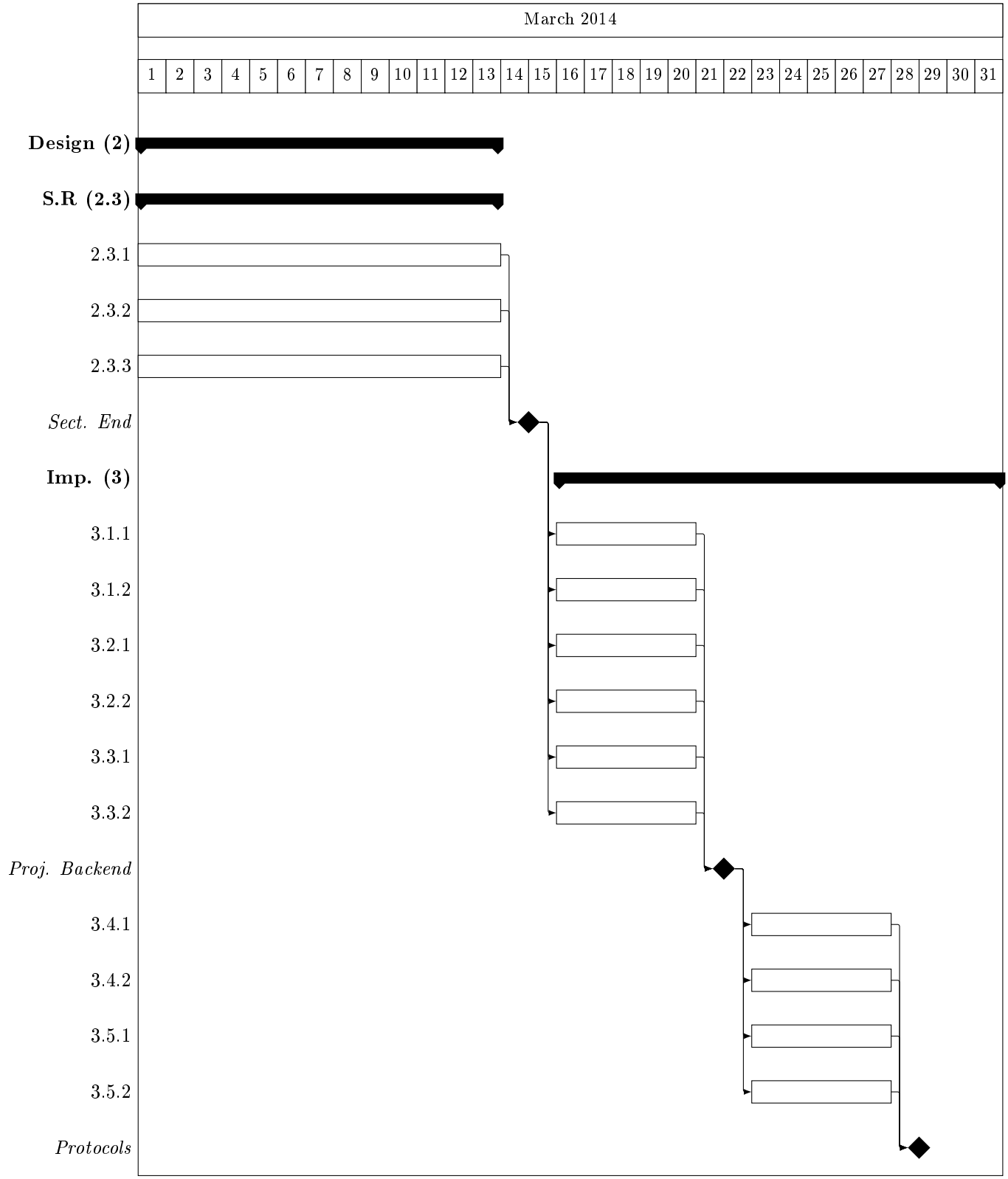
The following is Gantt Chart of the project. It is developed from looking at overall requirements of the project, and to act as a base for us to follow when developing the project.

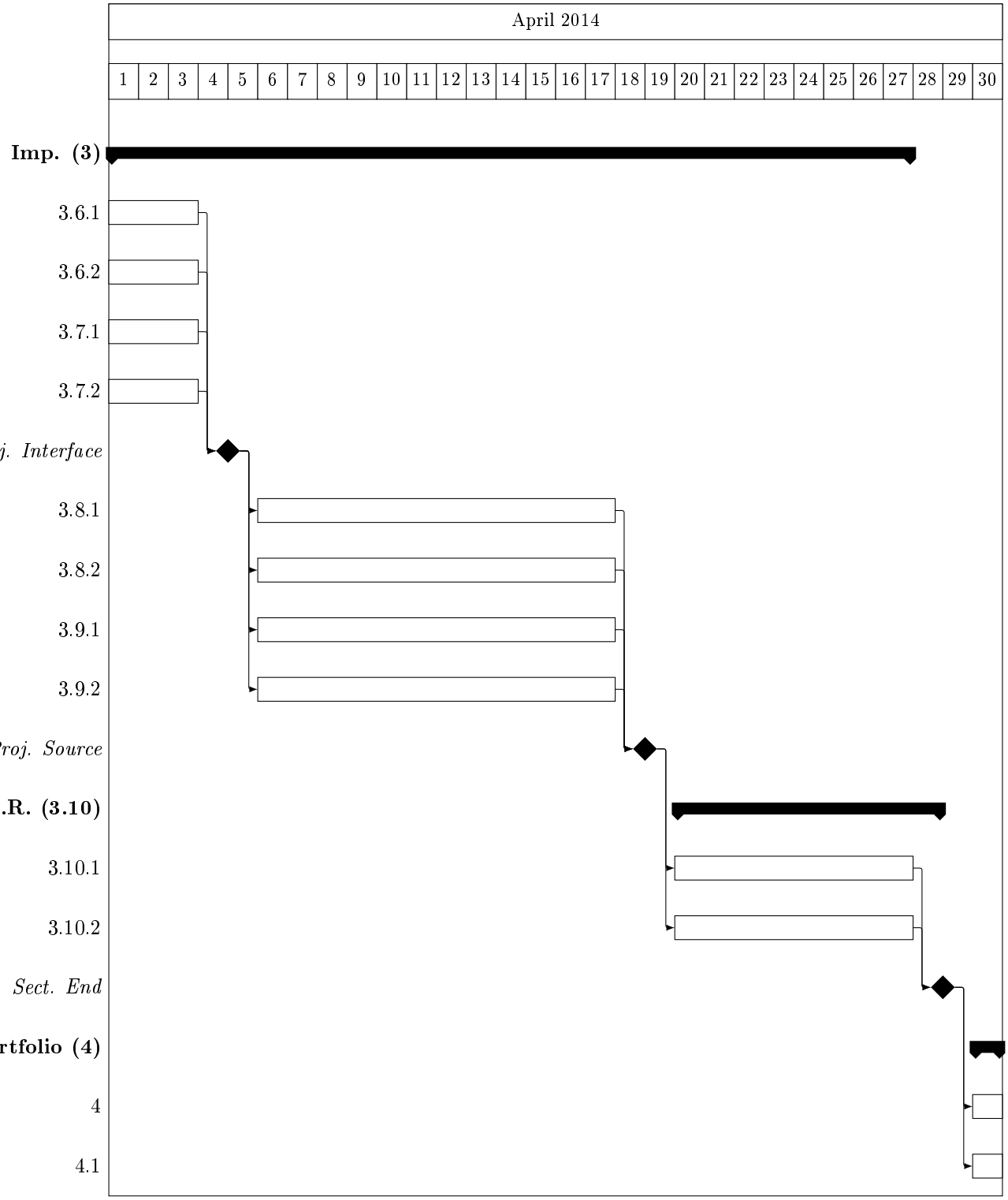
It is a graphical representation of the Task List documented prior to the charts. They have been split up either monthly or bi-monthly basis to allow acceptable formatting, due to differing workloads between months.

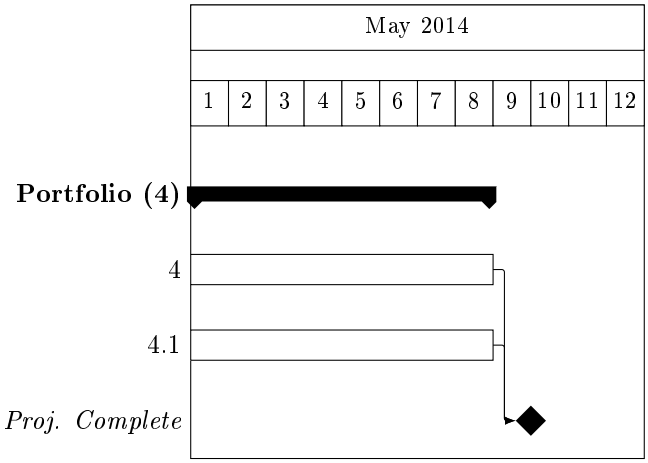
February is the most work intensive as it is providing the foundations of the project and is therefore detailed to reduce the risk of the project failing. April is the most lax of the working months (apart from May being project clean up) as the group members will be concentrating on the larger tasks as opposed to diverting attention between four or five tasks each, like in the Planning stage of the project.


February 2014
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Planning (1)
1.1
1.2
1.3
1.4
1.5
1.6
1.7
Scope
1.8.1
1.8.2
1.8.3
1.8.4
Case Studies
Technical Data
1.10
1.11
1.12
1.13
1.15
1.16
Planning Finished


February 2014
15
16
17
18
19
20
21
22
23
24
25
26
27
28
Design (2)
Research (2.1)
2.1.1
2.1.2
2.1.3
Research
sign Documentation(2.1)
2.2.1
2.2.2
2.2.3
2.2.4
2.2.5
2.2.6
2.2.7
2.2.8
2.2.9
2.2.10
2.2.11
2.2.12
2.2.13
Design Docs


March 2014
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Design (2)
S.R (2.3)
2.3.1
2.3.2
2.3.3
Sect. End
Imp. (3)
3.1.1
3.1.2
3.2.1
3.2.2
3.3.1
3.3.2
Proj. Backend
3.4.1
3.4.2
3.5.1
3.5.2
Protocols


April 2014
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
Imp. (3)
3.6.1
3.6.2
3.7.1
3.7.2
j. Interface
3.8.1
3.8.2
3.9.1
3.9.2
roj. Source
.R. (3.10)
3.10.1
3.10.2
Sect. End
rtfolio (4)
4
4.1


May 2014
1
2
3
4
5
6
7
8
9
10
11
12
Portfolio (4)
4
4.1
Proj. Complete

Chapter 18

Risk Assessment

18.1 Parallel Tasks

A big concern for any project is the amount of tasks that will be performed simultaneously [3]. 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 a Gantt chart 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.

18.2 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.

18.3 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 near the end of the deadline range to perform them.

18.4 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 [21]. 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.

18.5 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 [14]. 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. Assuming 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 [21]. All of these can amount into an increase in workload, or a decrease if the targets haven't been properly defined.

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.

18.6 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.
- Stakeholders becoming disillusioned: 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.
- Quality Risk: Stakeholders may give ambiguous input both accidentally or on purpose, depending whether the Stakeholder wants the project to fail or not [21].

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 [6].

18.7 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 [23]. 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 [13]. 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.

18.8 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 [21]

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.

18.9 Requirements

Requirements are not just a list of functional needs and wants but also the constraints on the project as well. However, there 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 [17].

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.

18.10 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.

18.11 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 which requires that you must provide encryption keys under certain circumstances to the UK authorities [4][22]. In the USA 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 oneâĂŹs mind

and is therefore testimonial.' [27]

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.

18.12 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.

18.13 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.

18.14 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.

Part II

Design

Chapter 19

Architecture

19.1 Network Architecture

Turtlenet is a centralized service, whereby a large number of clients connect to a single server which provides storage and facillitates communication between clients.

Due to the inherantly limited network size (5-50K users per server depending on percentage of active participants vs consumers and local internet speeds) we reccomend that servers serve a particular interest group or geographic locality.

Clients send messages to, and only to, these central servers. Due to the fact that all messages (except CLAIM messages, see client-server/client-client protocols for details) are encrypted the server does not maintain a databse, it cannot; rather clients each maintain their own local databased populated with such information to which they have been granted access.

When a client wishes to send a message to a person they encrypt the message with the public key of the recipient[1] and upload it to the server. It is important to note that all network connections are performed via Tor.

When a client wishes to view messages sent to them, they download all messages posted to the server since they last downloaded all messages from it and attempt to decrypt them all with their private key; those messages the client successfully decrypts (message decryption/integrity is verified via SHA256 hash) where intended for it and parsed. During the parsing of a message the sender is determined by seeing which known public key can verify the RSA signature.

Due to the nature of data storage in client-local databases, all events and data within the system must be represented within these plaintext messages. This is achieved by having multiple types of messages (see client-client protocol).

[1] using RSA/AES, see protocol for details

19.2 System Architecture

The system has a number of modules which interact with one another via strictly defined interfaces. Each module has one function, and interacts as little as possible with the rest of the system. The modules and their interactions are shown below. NB: a->b denotes that data passes from module a to module b, and a<->b similarly denotes that data passes both from a to b and from b to a.

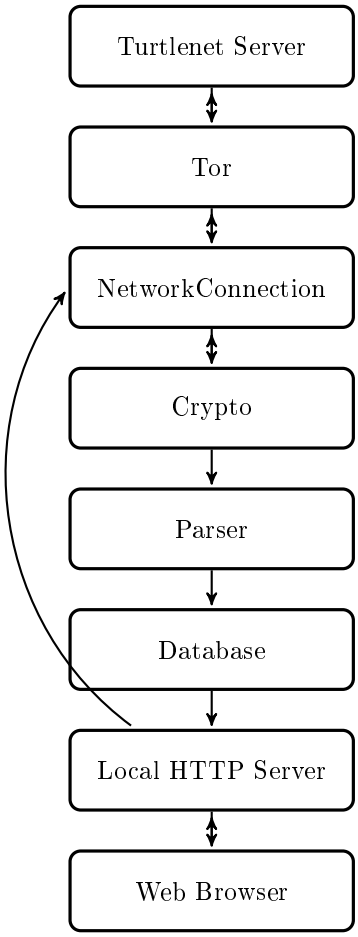


Figure 19.1: Module Interaction

Chapter 20

Data Flow Diagram

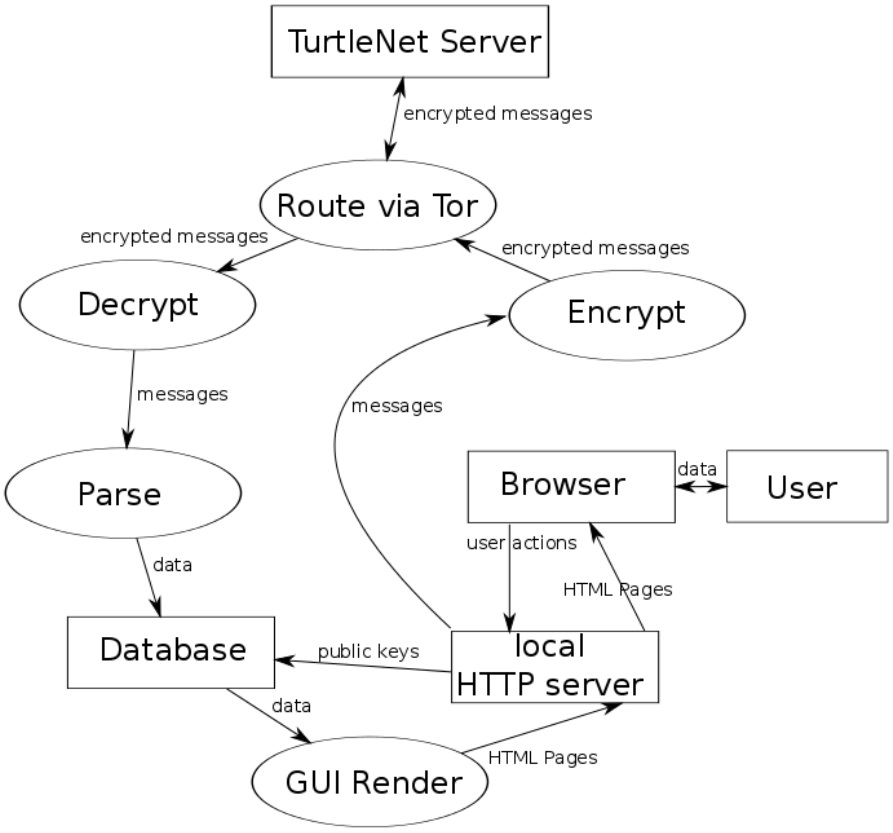
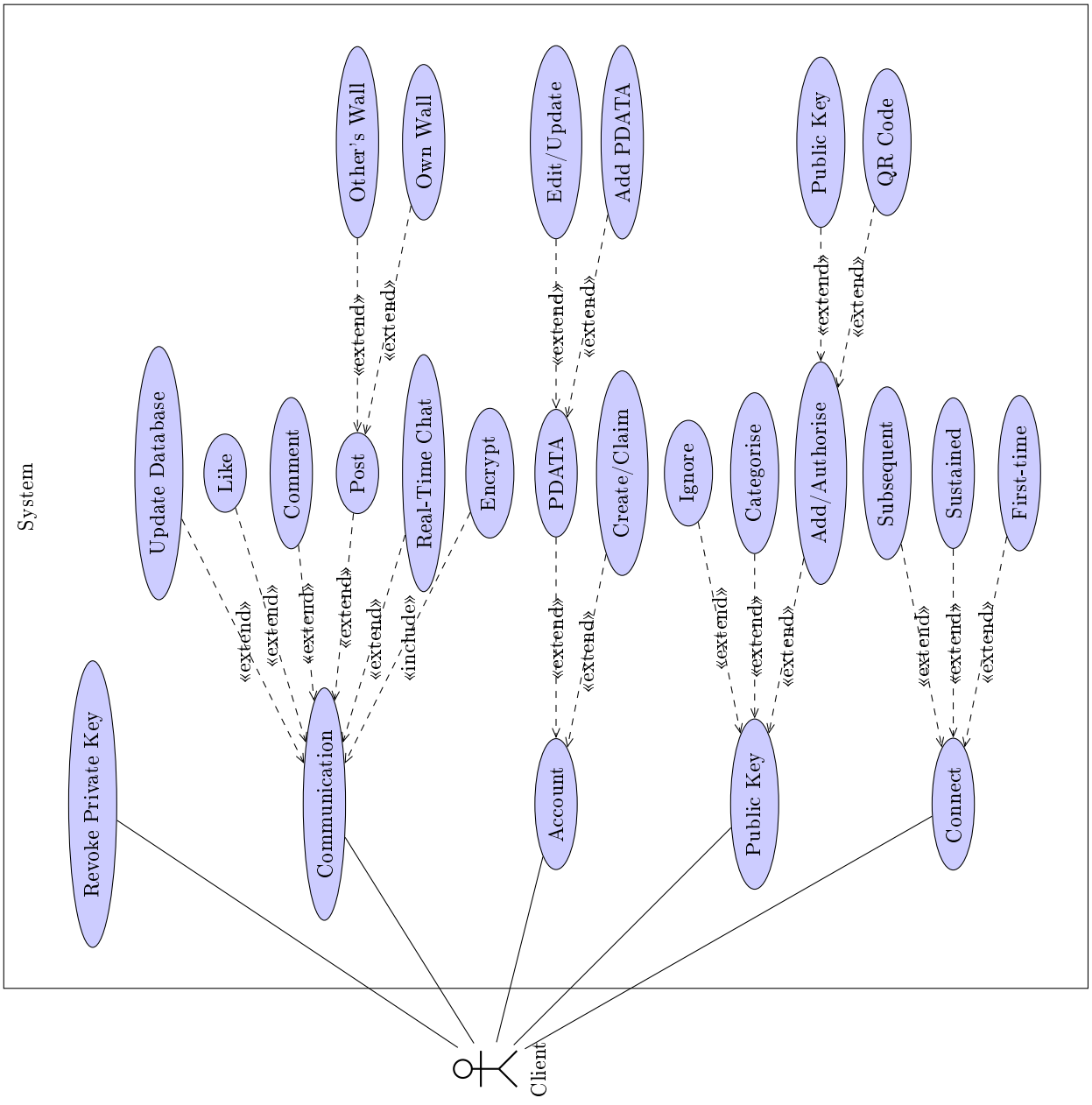
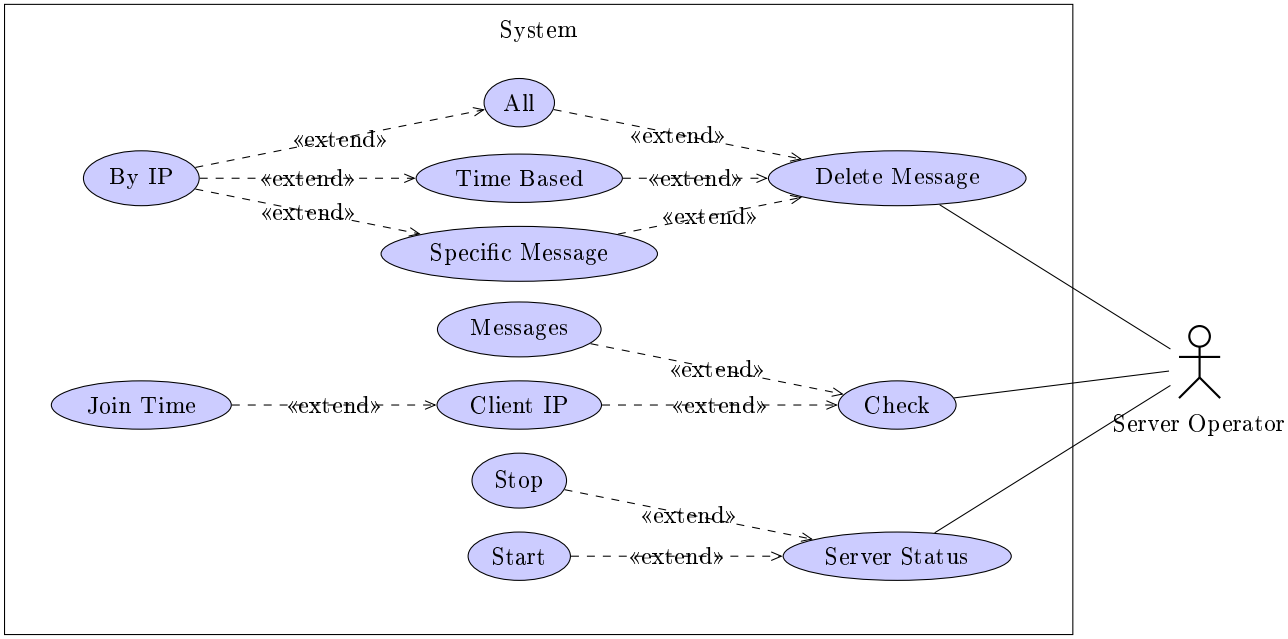


Figure 20.1: Data Flow Diagram

Chapter 21

Use Case Diagrams


System
Update Database
Like
Comment
Post
Other's Wall
Own Wall
Real-Time Chat
Encrypt
Edit/Update
Add PDATA
PDATA
Create/Claim
Ignore
Categorise
Add/Authorise
Public Key
QR Code
Subsequent
Sustained
First-time
Revoke Private Key
Communication
Account
Public Key
Connect
«extend»
«include»
Client


System
All
By IP
Time Based
Delete Message
Specific Message
Messages
Join Time
Client IP
Check
Stop
Start
Server Status
«extend»
Server Operator

Chapter 22

Protocol

22.1 High Level Summary of Protocol

ty 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). 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. It's worth noting that messages from before you connected for the first time are now downloaded because they can not have been sent to you (with a complient client) if someone retroactivly grants you permission to view something they publish it as a new message with an old timestamp; the sole exception to this is when you connect using a new device, in which case all messages since you first claimed a name will be downloaded.

Connecting subsequently The client requests every message stored on the server since the last time they connected up to the present. Decryptable messages are used to update the local DB, others are discarded.

Continued connection During a session the client requests updates from the server every

0.5-5 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 username (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. Public key transferal will be performed via exchaning plaintext base64 encoded strings, or QR codes. The user will be prompted, after retrieving the username of the user, to catagorise them.

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, commenting and liking may be requested by sending a EPOST/CMNT/LIKE message to the friend (upon whose wall/post you are posting, commenting or liking), when that friend logs in they will recieve your request 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. Given that authentication is entirely based on crypto signatures it doesn't matter that your friend relays the message.This is required because it is impossible for one to know who is able to see the persons wall, post, or comment upon which you seek to post, like, or comment.

22.2 Client-Server Protocol

The client-server architecture is neccessarily simple.

The client connects to the server, sends a single command, receives the servers response and then disconnects. The following shows commands sent by the client, and the servers action in response.

command	purpose	servers action
t	get the server time	sends back the current time (unix time in milliseconds)
s *utf-8_text*	send messages	the text sent is stored on the server
get *ms_unix_time*	get new messages	every message stored since the given time is sent
c *utf-8_text*	claim a username	the text sent is stored on the server, with a special filename

Table 22.1: Client-Server Protocol

Every command is terminated with a linefeed. Every response from the server will be terminated with a linefeed. The last line sent by the server will always be "s" for success, or "e" for failure (this is ommitted from the above table).

CLAIM messages (sent with c) will be parsed by the Message class and the username extracted for use in a filename. The filename of claim messages is as follows <*unix_time_in_ms*>_<*username*>; the filename of all other messages is as follows <*unix_time_in_ms*>_<*SHA256_hash*>.

22.3 Client-Client Protocol

22.4 Summary

sider computational
of seperating RSA
der and AES message

All client-client communication is mediated by the server. When one client wishes to send a message to another it encrypts the message with the public key associated with the recipient and uploads it to the server. When one client wishes to recieve a message it downloads all new messages from the server and parses those it can decrypt. This is performed in order to hide who recieves a message. All messages except CLAIM messages are encrypted. Multiple recipients imply multiple messages being uploaded, this is taken for granted in the text which follows.

22.5 Message Formatting

22.5.1 Unecrypted Messages

Messages have a command (or type), which specifies the nature of the message; messages have content, which specifies the details of the message; messages have an RSA signature, which authenticates the message; messages have a timestamp, which dates the message down to the millisecond, the time format is unix time in milliseconds.

Messages are represented external to the system as utf-8 strings, and internally via the Message class. The string representation is as follows:

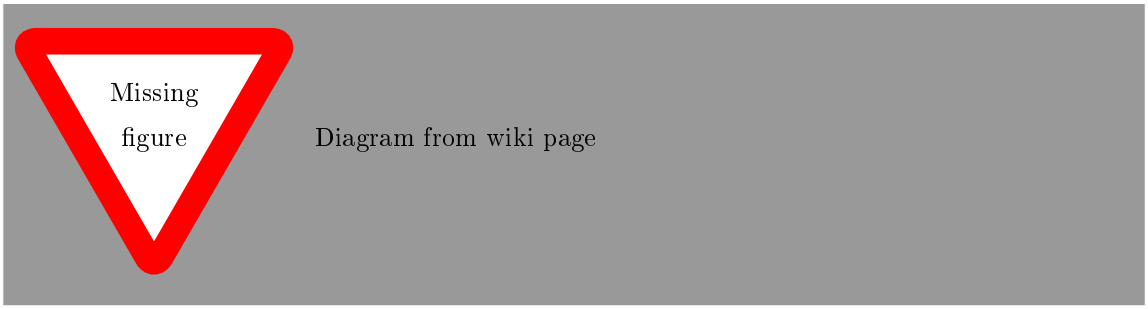
$$<command>\backslash<signature>\backslash<content>\backslash<timestamp>$$

Backslashes are literal, angle brackets denote placeholder values where data specific to a message is placed.

An example follows:

$$\text{POST}\backslash<signature>\backslash\text{Hello, World!}\backslash 1393407435547$$

backslashes in message content are escaped with another backslash, signatures are base64 encoded SHA256/RSA signatures of the content of the message concatenated with a decimal string representation of the timestamp. All text is encoded in UTF-8.



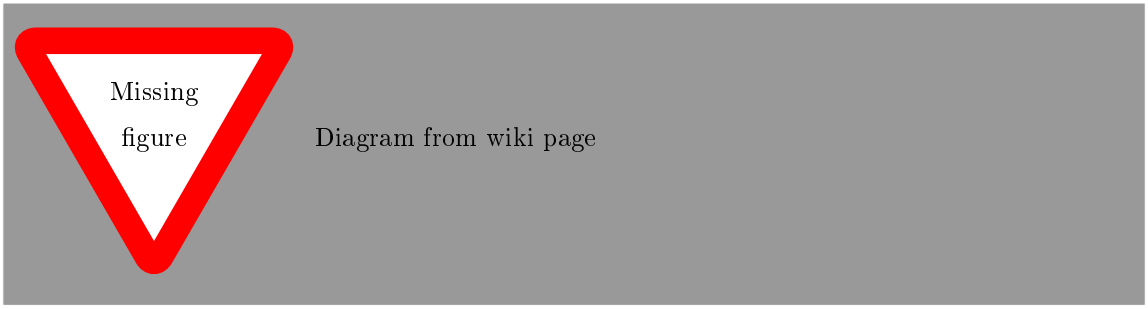
22.5.2 Encrypted Messages

Encrypted messages contain the AES IV's; the RSA encrypted AES key; and the AES encrypted message.

Messages are encrypted by encoding the entire message to be sent with UTF-8; encrypting the message with a randomly generated AES key; encrypting the AES key with RSA; encoding the RSA encrypted AES key in base64; encoding the (random) AES initialization vectors in base64 and concatenating these three parts with a backslash between each. The format follows:

$$<AES\ IV>\backslash<RSA\ encrypted\ random\ AES\ key>\backslash<AES\ encrypted\ message>$$

Backslashes are literal, angle brackets denote placeholder values where data specific to a message is placed.



22.6 Claiming a Username

Each user (keypair) should claim one username. Uniqueness is enforced by the server, and so not relied upon at all. Usernames are useful because public keys are not human readable. In order to claim a username, one must sent an unencrypted CLAIM message to the server. The format follows:

CLAIM\<*signature*>\<*username*>\<*timestamp*>

22.7 Revoking a Key

If a users private key should be leaked, then they must be able to revoke that key. This is done by sending a REVOKE message to the server. All content signed by the private key after the stated time will be flagged as untrusted. The format follows:

REVOKE\<*signature*>\<*time*>\<*timestamp*>

22.8 Profile Data

Users may wish to share personal details with certain people, they may share this information via profile data. Profile data is shared using PDATA messages. A PDATA message contains a list of fields, followed by a colon, followed by the value, followed by a semicolon. The format follows:

PDATA\<*signature*>\<*values*>\<*timestamp*>

The format for values follows:

<*field*>: <*value*>; ...

An example follows:

PDATA\<*signature*>\name:Luke Thomas;dob:1994;\<*timestamp*>

22.9 Inter-User Realtime Chat

Users can chat in in real time, this by achieved by sending a CHAT message to all people you wish the include in the conversation. This message includes a full list of colon delimited public keys involved in the chat. The format follows:

CHAT\<*signature*>\<*keys*>\<*timestamp*>

The format for keys follows:

<*key*>: <*another_key*>...

An example follows:

CHAT\<*signature*>\<*key1*>:<*key2*>\<*timestamp*>

Following the establishment of a conversation, messages may be added to it with PCHAT messages, the format follows:

PCHAT\<*signature*>\<*conversation*>:<*message*>\<*timestamp*>

Whereby <*conversation*>denotes the signature present on the establishing message. An example follows:

PCHAT\<*signature*>\9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08:First!\<*timestamp*>

22.10 Posting to own wall

When a user posts to their own wall they upload a POST message to the server of the following format.

POST\<*signature*>\<*message*>\<*timestamp*>

The format of message is merely UTF-8 text, with backslashes escaped with blackslashes.
An example follows which contains the text "Hello, World!", a newline, "foo \bar\baz":

POST\<*signature*>\Hello, World!
foo\\bar\\baz\<*timestamp*>

22.11 Posting on anothers wall

A user may request to post on a friends wall by sending them an FPOST message, the poster may not decide who is able to view the message. The format is identical to that of a POST message, except for the command and singular recipient. An example follows:

FPOST\<*signature*>\Hello, World!\<*timestamp*>

Upon recipt of an FPOST message the friend is prompted by the client to choose whether or not to display it, and if so who may view it. Once this is done the friend reposts the message with the command changed to POST instead of FPOST as they would post anything to their own wall. This works because authentication is entirely based on RSA signatures so in copying the original signature the friend may post as the original author provided they don't alter the message (and thus its hash and required signature).

22.12 Commenting

Commenting works simiarly to posting on anothers wall, so an explanation of details of how it occurs is not provided (see prior section). The only difference is the format of a CMNT message from an FPOST message. The format of a CMNT message is as follows:

CMNT\<*signature*>\<*hash*>: <*comment*>\<*timestamp*>

Where <*hash*>denotes the hash of the post or comment being commented upon. An example comment follows:

CMNT\<*signature*>\9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08:
Yeah, well, that's just like, your opinion, man.\<*timestamp*>

22.13 Liking

Like messages are identical to comments except for the command and the the fact that no ":<*comment*>" follows the hash. An example like follows:

LIKE\<*signature*>\9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08\<*timestamp*>

Chapter 23

Class Interfaces

23.1 Class Interfaces

The following is a description of the public functions of all public classes. Many classes have inner private classes they use for conveinience, however to simplify interaction between parts of our system ('modules') we have very few conveinience classes.

Reoconcile return type
with stated public clas

specify what's static

function	description
void main()	starts the server

Table 23.1: Server

function	description
void main()	constructs and starts all nececary classes and threads, runs the main loop

Table 23.2: Client

go over DB interface w
GUI guys and aishiah

function	description
NetworkConnection()	Constructs a NetworkConnection and connects to the given URL (through tor)
void run()	periodically download new messages until asked to close, downloaded messages are stored in a FIFO buffer
void close()	kills the thread started by run()
boolean hasMessage()	return true if there is a message in the buffer, false otherwise
String getMessage()	return the oldest message in the buffer
boolean claimName()	claim a given username, returns true on success, false otherwise
void revokeKeypair()	revokes your keypair
void pdata()	adds or updates profile infomration
void chat()	begins or continues a conversaion
void post()	post a message to your wall
void fpost()	post a message to a friends wall
void comment()	comment on a comment or post
void like()	like a comment or post

Table 23.3: NetworkConnection

23.2 Class Diagram

function	description
boolean keysExist()	return true if the user has a keypair, false otherwise
void keyGen()	generate a keypair for the user
PublicKey getPublicKey()	returns the users public key
PrivateKey getPrivateKey()	returns the users private key
String sign()	returns an RSA signature of the passed string
boolean verifySig()	returns true if author signed msg, false otherwise
String encrypt()	returns an encrypted message constructed from the passed parameters
Message decrypt()	decrypts the passed string, returns the appropriate message, on failure a NULL message is returned
String base64Encode()	base64 encodes the passed data, returns the string
byte[] base64Decode()	base64 decodes the passed data, returns the byte[]
String encodeKey()	encodes a public key as a string, returns that string (X509)
PublicKey decodeKey()	decodes a public key encoded as a string, returns that public key(X509)
String hash ()	returns the SHA256 hash the the passed string as a hex string
int rand ()	returns a pseudorandom value <= max and >= min

Table 23.4: Crypto

function	description
void parse()	parses a sting message, records parsed data in the database

Table 23.5: Parser

function	description
PublicKey getPublicKey()	returns the public key associated with the username
String getUsername()	returns the username associated with a public key
Post[] getPosts()	returns posts posted with given timeframe (unix time in milliseconds)
Comment[] getComments()	returns all comments on the given post
Like[] getLikes()	returns all likes on the given post or comment
Conversation getConversation()	returns a given conversaion
addFriend()	adds the given friend to the DB
addCategory()	adds a new category to the DB
addToCategory()	adds a user to a category

Table 23.6: Database

function	description
GUIServer()	Constructs a GUIServer
void run()	continually updates the GUI from the DB
void close()	kills the GUIServer thread
boolean isRunning()	returns true if the GUIServer is running, false otherwise

Table 23.7: GUI

function	description
Message()	Consturcts a message with given data
Message parse()	parses the string representation of a message into a message
String toString()	creates a string representation of the message
String getCmd()	returns the type of message
String getContent()	returns the content of the message
String getSig()	returns the RSA signature on the message
long getTimestamp()	returns the timestamp on the message

Table 23.8: Message

Chapter 24

Pseudocode

List important functio
copy/paste from java a
edit

Chapter 25

Database

ge w/transaction de-

25.1 Database execution

In this section, we go through the execution methods of the database based on the transactions that have been carried out within the system. This also shows where the data is expected to roughly end up, however this will be explained in greater detail along with the diagrams which will be found later in this document.

Stakeholders and users have to be aware that due to lightweight database files are stored locally in each users' computer, there are a number of databases involve when the transactions are carried out. The reason why it is designed this way is to ensure and avoid any malicious activities conducted especially by the server.

25.1.1 User adds post, comment and event

When a user adds a content into Turtlenet such as posts, comments and events, the system is expected to capture these details and add them into its respective tables. The database system is expected to log the posts, comments and events by capturing the time and date when the transaction is carried out.

25.1.2 User creates and sends message to another user

As when the user creates a message then sends it, the database system is expected to store the message and log it by recording its date and time of which the message is sent. Other details like the reciever's user_id are inserted into the database well.

25.1.3 User sends a friend request to another user

When the user sends a request to others, this request will be sent and the details will be captured and recorded into the other user's local database file under the friend request table. This will be stored in this table until which the user decides to either accept or reject the invitation.

25.1.4 User receives a friend request from another user

Another situation with the friend request is when a user receives a friend request, this time the information such as the public key will be recorded into the user's local database until which the user decides to do something either accept or reject it.

25.1.5 A user adds a relation

When a user adds a relation, the details of this related user will be captured, such as his profile, and will be added into the users table. From then on, the user can see his relation's profile information.

25.1.6 User receives a message

As the user receives a message, it will be stored in the message table along with other details such as the date and time, and the sender's details.

25.1.7 User receives a friend request

The user will be notified when a friend request is sent. The details of the person who sends the request will be recorded in the database. The user has two options to deal with a friend request, either to accept or reject it. Once it is accepted, the profile details of the sender will be stored in the user's local database, same goes to the user's details store on the sender's local database.

25.2 Table layout of the database

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

Table 25.1: you

Name	Datatype	Key
username	VARCHAR(25)	PK
name	VARCHAR(30)	
birthday	DATE	
sex	VARCHAR(1)	
email	VARCHAR(30)	
public_key	VARCHAR(8)	

Table 25.2: user

Name	Datatype	Key
user_id	VARCHAR(8)	PK
username	VARCHAR(25)	
name	VARCHAR(30)	
birthday	DATE	
sex	VARCHAR(1)	
email	VARCHAR(30)	

Table 25.3: is_in_category

Name	Datatype	Key
is_in_id	INT(10)	PK
category_id	INT(10)	FK
user_id	INT(5)	FK

Table 25.4: category

Name	Datatype	Key
category_id	INT(10)	PK
name	VARCHAR(30)	

Table 25.5: private_message

Name	Datatype	Key
message_id	INT(10)	PK
from	VARCHAR(8)	
to	VARCHAR(8)	
content	VARCHAR(50)	
time	DATE	

Table 25.6: wall_post

Name	Datatype	Key
wall_id	INT(10)	PK
from	VARCHAR(8)	FK
to	VARCHAR(8)	FK
permission_to	VARCHAR(8)	FK
content	VARCHAR(50)	
time	DATE	

Table 25.7: has_comment

Name	Datatype	Key
comment_id	INT(10)	PK
post_id	INT(10)	FK
user_id	VARCHAR(8)	FK
comment_time	DATE	

Table 25.8: has_like

Name	Datatype	Key
like_id	INT(10)	PK
post_id	INT(8)	FK
user_id	VARCHAR(8)	FK

Table 25.9: events

Name	Datatype	Key
event_id	INT(10)	PK
title	VARCHAR(10)	
content	VARCHAR(40)	
from	VARCHAR(8)	FK
permission_allowed_to	VARCHAR(8)	FK

Table 25.10: friend_request

Name	Datatype	Key
public_key	VARCHAR(8)	PK
request_time	DATE	
request_react	VARCHAR(10)	

Table 25.11: login_logout_log

Name	Datatype	Key
log_id	INT(10)	PK
login_time	DATE	
logout_time	DATE	

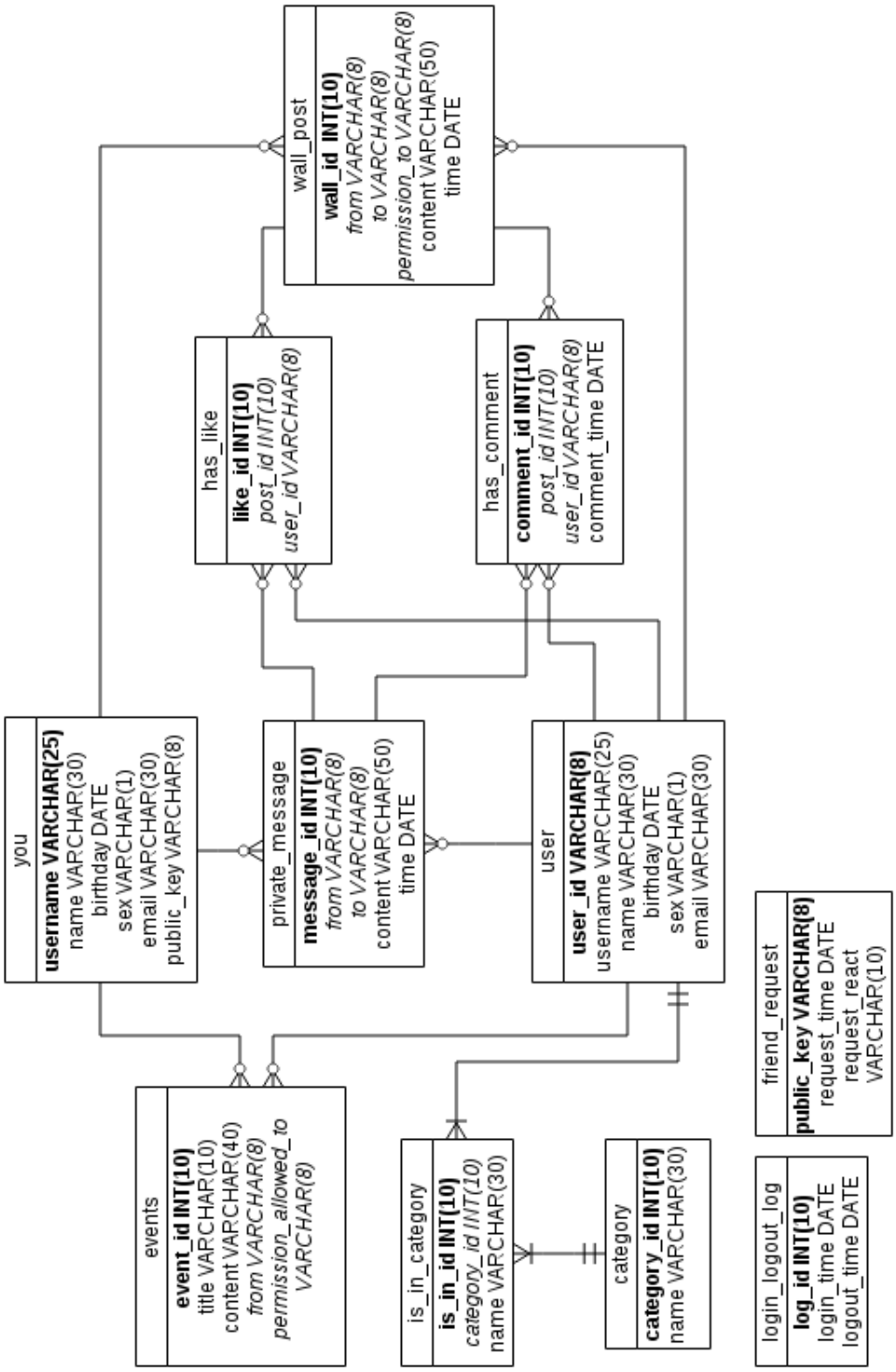


Figure 25.1: Database Entity Relationship diagram

Chapter 26

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

26.1 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 26.1: User enters his profile information

_id's and post_id's
local and don't exist
ide 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 26.2: 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 26.3: 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 26.4: user adds a new event

field	notes
comment_id	
post_id	
comment_from	
comment_time_ date	

Table 26.5: user adds a comment

field	notes
like_id	
post_id	
like_from	

Table 26.6: user likes a post

Chapter 27

User Interfaces

As a social network, the user interface design is of high importance, as a lot of users of the program will have little core system knowledge, and rely entirely on the user interface. As a result we have looked at a variety of options into designing which will be the best for the project.

27.1 Swing

Swing is the primary Java GUI toolkit, providing a basic standpoint for entry level interface designing. Introduced back in 1996, Swing was designed to be an interface style that required minimal changes to the applications code, providing the user with a pluggable look and feel mechanism. It has been apart of the standard java library for over a decade, which, as I will now explain, may not be to our benefit.

Swing, whilst an excellent language to begin with, and write simple applications in, is quite dated. As our group advisor put it when inquiring about what we would be coding the user interface in:

> "You should avoid Swing to prevent it looking like it was done in the seventies." - Sebastian Coope

Sebastian is not wrong either, as Swing does a very plain feel to it. "Fig ???" shows an old instant messaging system written with Swing by one of our team members. As you can see it is unlikely to appeal to the mass market with such visually plain appearance. This makes Swing, unlikely to be our GUI toolkit of choice, despite some of our members experience with it.

27.2 Abstract Window Toolkit

Abstract Window Toolkit (otherwise known as AWT), was another choice given that we are programming in Java, and synchronicity between the two would be an advantage. Whilst AWT retained some advantages such as its style blending in with each operating system it runs on, it is even older than Swing being Java's original toolkit, as per such making it redundant for this project.

rsa is 18 years older, w
going to need a better
son about how it's wie
for users, web tech is n
developed for this type
application, or somethi

27.3 Standard Widget Toolkit

Standard Widget Toolkit (otherwise known as SWT), is one of the more promising candidates so far given its look and up-to-date support packages. The latest stable release of SWT was only last year, and is capable of producing programs with a modern and professionally built appearance, as shown in "Fig ???".

Unlike both Swing and AWT, SWT is not provided by Sun Microsystems as a part of the Java platform. It is now provided and maintained by the Eclipse Foundation, and provided as a part of their widely used Eclipse IDE, something a lot of the team is familiar with.

27.4 GWT

GWT allows you to create HTML/Javascript based user interfaces for Java applications running locally. The interface is programmed in Java and then GWT creates valid HTML/Javascript automatically. A web server is required in order for Javascript events to be sent to the Java application.

The user can then interact with the system by pointing their web browser at localhost. This has the benefit of being familiar to novice users as most modern computer interaction is done within a web browser.

Another advantage of using GWT is the ability to alter the appearance of web pages using CSS. This facilitates the creation of a modern, attractive user interface that integrates nicely with current operating systems and software.

27.5 Javascript

It is possible to create the entire client application in Javascript and use a HTML/Javascript GUI. This approach removes the need for a local web server meaning the only software the user is required to run is a modern web browser.

Another advantage would be tight integration between the logic and interface elements of the client application and no risk of errors caused by using multiple programming languages.

One disadvantage of this approach is the difficulty in implementing the required security measures and encryption in Javascript. This can be remedied by using a Javascript library such as the Forge project which implements many cryptography methods.

The main disadvantage is that in this approach the server operator has complete control of the client the user uses. This is unacceptable because we're assuming that the server operator is seeking to spy on the user.

Chapter 28

QR

I've found a website that generates QR Codes - both professionally and otherwise for free. We could implement it into our prgram by having the program output the URL it gives you as because it's generated via URL, we should be able to store it in a string and then output either that or have an image viewer in the program to output the actual image, whichever is easiest for the user.

The website's create function: http://goqr.me/api/doc/create-qr-code/ The website's read function: http://goqr.me/api/doc/read-qr-code/ A test one I did: https://api.qrserver.com/v1/create-qr-code/?size=300x300&data=%3Ci'mThePublicKeyVariable%3E&format=svg

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

To the extent possible under law, Ballmer Peak has waived all copyright and related or neighboring rights to Turtlenet and Associated Documentation. This work is published from: United Kingdom.

B.1 Statement of Purpose

The laws of most jurisdictions throughout the world automatically confer exclusive Copyright and Related Rights (defined below) upon the creator and subsequent owner(s) (each and all, an "owner") of an original work of authorship and/or a database (each, a "Work").

Certain owners wish to permanently relinquish those rights to a Work for the purpose of contributing to a commons of creative, cultural and scientific works ("Commons") that the public can reliably and without fear of later claims of infringement build upon, modify, incorporate in other works, reuse and redistribute as freely as possible in any form whatsoever and for any purposes, including without limitation commercial purposes. These owners may contribute to the Commons to promote the ideal of a free culture and the further production of creative, cultural and scientific works, or to gain reputation or greater distribution for their Work in part through the use and efforts of others.

For these and/or other purposes and motivations, and without any expectation of additional consideration or compensation, the person associating CC0 with a Work (the "Affirmer"), to the extent that he or she is an owner of Copyright and Related Rights in the Work, voluntarily elects

to apply CC0 to the Work and publicly distribute the Work under its terms, with knowledge of his or her Copyright and Related Rights in the Work and the meaning and intended legal effect of CC0 on those rights.

B.2 Copyright and Related Rights

A Work made available under CC0 may be protected by copyright and related or neighboring rights ("Copyright and Related Rights"). Copyright and Related Rights include, but are not limited to, the following:

1. the right to reproduce, adapt, distribute, perform, display, communicate, and translate a Work;
2. moral rights retained by the original author(s) and/or performer(s);
3. publicity and privacy rights pertaining to a person's image or likeness depicted in a Work;
4. rights protecting against unfair competition in regards to a Work, subject to the limitations in paragraph 4(a), below;
5. rights protecting the extraction, dissemination, use and reuse of data in a Work;
6. database rights (such as those arising under Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, and under any national implementation thereof, including any amended or successor version of such directive); and
7. other similar, equivalent or corresponding rights throughout the world based on applicable law or treaty, and any national implementations thereof.

B.3 Waiver

To the greatest extent permitted by, but not in contravention of, applicable law, Affirmer hereby overtly, fully, permanently, irrevocably and unconditionally waives, abandons, and surrenders all of Affirmer's Copyright and Related Rights and associated claims and causes of action, whether now known or unknown (including existing as well as future claims and causes of action), in the Work (i) in all territories worldwide, (ii) for the maximum duration provided by applicable law or treaty (including future time extensions), (iii) in any current or future medium and for any number of copies, and (iv) for any purpose whatsoever, including without limitation commercial, advertising or promotional purposes (the "Waiver"). Affirmer makes the Waiver for the benefit of each member

of the public at large and to the detriment of Affirmer's heirs and successors, fully intending that such Waiver shall not be subject to revocation, rescission, cancellation, termination, or any other legal or equitable action to disrupt the quiet enjoyment of the Work by the public as contemplated by Affirmer's express Statement of Purpose.

B.4 Public License Fallback

Should any part of the Waiver for any reason be judged legally invalid or ineffective under applicable law, then the Waiver shall be preserved to the maximum extent permitted taking into account Affirmer's express Statement of Purpose. In addition, to the extent the Waiver is so judged Affirmer hereby grants to each affected person a royalty-free, non transferable, non sublicensable, non exclusive, irrevocable and unconditional license to exercise Affirmer's Copyright and Related Rights in the Work (i) in all territories worldwide, (ii) for the maximum duration provided by applicable law or treaty (including future time extensions), (iii) in any current or future medium and for any number of copies, and (iv) for any purpose whatsoever, including without limitation commercial, advertising or promotional purposes (the "License"). The License shall be deemed effective as of the date CC0 was applied by Affirmer to the Work. Should any part of the License for any reason be judged legally invalid or ineffective under applicable law, such partial invalidity or ineffectiveness shall not invalidate the remainder of the License, and in such case Affirmer hereby affirms that he or she will not (i) exercise any of his or her remaining Copyright and Related Rights in the Work or (ii) assert any associated claims and causes of action with respect to the Work, in either case contrary to Affirmer's express Statement of Purpose.

B.5 Limitations and Disclaimers

1. No trademark or patent rights held by Affirmer are waived, abandoned, surrendered, licensed or otherwise affected by this document.

2. Affirmer offers the Work as-is and makes no representations or warranties of any kind concerning the Work, express, implied, statutory or otherwise, including without limitation warranties of title, merchantability, fitness for a particular purpose, non infringement, or the absence of latent or other defects, accuracy, or the present or absence of errors, whether or not discoverable, all to the greatest extent permissible under applicable law.

3. Affirmer disclaims responsibility for clearing rights of other persons that may apply to the Work or any use thereof, including without limitation any person's Copyright and Related Rights in the Work. Further, Affirmer disclaims responsibility for obtaining any necessary consents, permissions or other rights required for any use of the Work.

4. Affirmer understands and acknowledges that Creative Commons is not a party to this document and has no duty or obligation with respect to this CC0 or use of the Work.

B.6 Included Works

We did not write or create the following:

- writeup/latex/tikz-uml.sty
- writeup/latex/todonotes.sty
- writeup/latex/ulem.sty
- writeup/images/appendicies/licence.png (CC0 licence logo)
- The CC0 licence text
- client/web_interface_mockup/jquery.js
- client/web_interface_mockup/turtles.ttf

look into legality of distribution

look into legality of distribution

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
escape backslashes in message content
chang all references to ascii text to UTF-8 text

C.2 Requirements Weeks 1-3

1. Project Desc.

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

2. Statement of Deliverables

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

- **COMPLETE** User view and requirements (Luke)
- **COMPLETE** System requirements (Luke)
- **COMPLETE** Transaction requirements (Aishah)

3. Project and Plan

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

4. Bibliography

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

C.3 Design Weeks 4-X

- **DRAFTED** Use Case Diagram (Mike)
- **DRAFTED** Data Dictionary (Mike)
- **NOT IN PDF** Mobile GUI Design (Leon)
- **NOT IN PDF** Sequence Diagram (Leon)
- **NOT IN PDF** HTML GUI Design (Louis)
- **DRAFTED** DB Design (Aishah)
- **DRAFTED** Transaction Design (Aishah)

- **INCOMPLETE** Server GUI Design (Peter)
- **DRAFTED** Class Interfaces (Luke)
- **DRAFTED** Protocol (Luke)
- **DRAFTED** Architecture (Luke)
- **DRAFTED** Data Flow Diagrams (Luke)
- **NOT IN PDF** Pseudocode (Luke)
- **INCOMPLETE** Class Diagram (???)

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] BBC. Azhar Ahmed convicted of offensive Facebook message. English. The BBC. Sept. 2012. URL: http://www.bbc.co.uk/news/uk-england-leeds-19604735 (visited on 02/12/2014).

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

[3] 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).

[4] The Crown. Regulation of Investigatory Powers Act 2000. English. July 2000. URL: http://www.legislation.gov.uk/ukpga/2000/23/section/50 (visited on 02/12/2014).

[5] Wei Dai. Crypto++ 5.6.0 Benchmarks. English. Mar. 2009. URL: http://www.cryptopp.com/benchmarks.html (visited on 02/13/2014).

[6] A. Dcosta and M. Gundlach. Stakeholders Risk Management - Project Risk Assessment Approach. 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).

[7] 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).

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

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

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

[11] J. Scahill Glenn Greenwald. The NSAâĂŹs Secret Role in the U.S. Assassination Program. English. The Intercept. Feb. 2014. URL: https://firstlook.org/theintercept/article/2014/02/10/the-nsas-secret-role/ (visited on 02/12/2014).

[12] K.L. Huff. WebPG for Mozilla. English. Jan. 2013. URL: https://addons.mozilla.org/en-US/firefox/addon/webpg-firefox/ (visited on 02/13/2014).

[13] 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).

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

[15] The Tor Project Inc. Tor Project: Overview. English. Feb. 2014. URL: https://www.torproject.org/about/overview.html.en (visited on 02/13/2014).

[16] J. Garside J. Ball L. Harding. BT and Vodafone among telecoms companies passing details to GCHQ. English. The Guardian. Aug. 2013. URL: http://www.theguardian.com/business/2013/aug/02/telecoms-bt-vodafone-cables-gchq (visited on 02/12/2014).

[17] J.P.Lewis and U. 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).

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

[19] C. Labovitz. Libya Firewall Begins to Crumble? English. Feb. 2011. URL: http://www.monkey.org/~labovit/blog/ (visited on 02/27/2011).

[20] 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).

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

[22] 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).

[23] 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).

[24] J. Risen. N.S.A. Gathers Data on Social Connections of U.S. Citizens. English. The New York Times. Sept. 2013. URL: http://www.nytimes.com/2013/09/29/us/nsa-examines-social-networks-of-us-citizens.html?_r=1&pagewanted=all& (visited on 02/12/2014).

[25] 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).

[26] Tom Ritter. The Differences Between Onion Routing and Mix Networks. English. Apr. 2013. URL: http://ritter.vg/blog-mix_and_onion_networks.html (visited on 02/13/2014).

[27] P. 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).

[28] 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).

[29] 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).

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

[31] various. Global surveillance disclosures (2013âĂŞpresent). English. Wikipedia. Feb. 2014. URL: https://en.wikipedia.org/wiki/Global_surveillance_disclosures_%282013%E2%80%93present%29 (visited on 02/12/2014).

[32] P. Wiseman. Cracking the 'Great Firewall' of China's Web censorship. English. USA TODAY. Apr. 2008. URL: http://abcnews.go.com/Technology/story?id=4707107 (visited on 04/24/2008).