# OMiN - An Opportunistic Microblogging Network

Neil Wells & Tristan Henderson

## ABSTRACT

OMiN is a pocket switched network running on smartphones. It allows users to send and receive messages without using any global infrastructure such as the internet. Smartphones in close proximity to each other pass on messages via Bluetooth. Steps have been taken to secure the network and protect it from known attack vectors. A variation of the PROPHET routing algorithm is used to effectively route messages.

# DECLARATION

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated.

The main text of this project report is **NN,NNN* TODO** words long, including project specification and plan.

In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.

# CONTENTS

# 1 INTRODUCTION

**TODO**

# 2 USER MANUAL

## 2.1 SOURCE CODE

Source code is stored using the school's Mercurial service at http://ndw.hg.cs.st-andrews.ac.uk/sh-proj and on GitHub at https://github.com/neilw4/OMiN. To download the source code, use the command

```
hg clone http://ndw.hg.cs.st-andrews.ac.uk/sh-proj
```

or

```
git clone http://github.com/neilw4/OMiN.git
```

## 2.2 PROJECT LAYOUT

The project is split into three different modules, each build using Gradle:

- The *app* module contains the Android app to be installed on every node.
- The *pkg* module is the central authentication server, which runs on the school's host server via CGI.
- The *crypto* module is a library of cryptography functions used by both the app and authentication server.

## 2.3 BUILDING

The android project uses the Android SDK version 21. Executing the following command from the project directory will build everything, downloading libraries and build scripts if necessary:

```
./gradlew build
```

If the android sdk location cannot be found, create a file called *local.properties* in the project folder containing the line "*sdk.dir=<sdk_location>",* where *<sdk_location>* is the location of the Android SDK.

The binaries will now be in the following locations:

- The main app will be located at app/build/outputs/apk/app-debug.apk
- The cryptography library will be at crypto/build/libs/crypto.jar
- The authentication server will be at pkg/ build/libs/pkg.jar and can be executed using the CGI script at pkg/omin.cgi

## 2.4 INSTALLATION

### *2.4.1 Authentication Server*

To run the authentication server, configure a web server to run the *omin.cgi* script in the *pkg* directory. The Android app will have to be modified to use the new server location and master

public key. The server stores private information such as the master keys in the working directory, so it is essential that the web server cannot serve these files (e.g. by creating a separate CGI script in the public directory of the web server to call the authentication script in a non-public directory).

### *2.4.2 Android App*

# TODO: APP STORE

The app can be installed from the app store or executing the following command:

```
./gradlew installDebug
```

## 2.5 USE

# TODO

# 3 CONTEXT SURVEY

The following provides brief summary of opportunistic networks and the current state-of-the-art in opportunistic network technology. Only the most relevant subjects will be addressed in order to give the reader sufficient background information to fully understand the project.

## 3.1 OPPORTUNISTIC NETWORKS

An opportunistic network is a network where connections between nodes are sparse and a direct path from source to destination is rarely possible. For example, a common form of opportunistic network (and the form we will focus on) is the Pocket Switched Network (PSN) - a network of smartphones carried around by people. Connections are made between smartphones in close proximity using a short range protocol such as Bluetooth. Because of the predictable nature of human behaviour, much research has been done to improve PSN algorithms.

Opportunistic network nodes must store messages and forward them to other nodes where possible. Messages often take a significant amount of time to reach their destination: this makes it much harder to solve problems that have been solved in conventional connected networks (security, routing etc.), which assume near-instant message transfer.

## 3.2 SIMILAR PROJECTS

There are a number of projects utilising opportunistic networks and similar technologies. I have listed the most relevant ones here.

### *3.2.1 Haggle*

Haggle (http://www.haggleproject.org) (1) - a pocket switched network designed to run on smartphones - is one of the largest opportunistic networks. There are implementations for a number of clients including Android (play.google.com/store/apps/details?id=org.haggle.kernel) and Windows Mobile.

By monitoring use of the platform, the authors discovered trends in inter-contact times and contact durations, showing that conventional opportunistic routing algorithms are poorly suited to real world pocket switched networks (2).

### *3.2.2 FireChat*

FireChat (opengarden.com/firechat) is a smartphone application used for off-the-grid messaging between nearby users. It has been used to circumvent government restrictions in Iraq (http://www.theguardian.com/technology/2014/jun/24/firechat-updates-as-40000-iraqis-download-mesh-chat-app-to-get-online-in-censored-baghdad) and during the Hong Kong protests (http://www.theguardian.com/world/2014/sep/29/firechat-messaging-app-powering-hong-kong-protests).

However, the app mostly relies on an internet connection, and its simple protocol is insecure (http://breizh-entropy.org/~nameless/random/posts/firechat_and_nearby_communication) and unable to implement the store-and-forward functionality of a proper opportunistic network.

OMiN will be a secure alternative to firechat which does not rely on an internet connection.

### 3.2.3 SWIM

The Shared Wireless Infostation Model (SWIM) is a proposed opportunistic network to monitor whales (3). Small nodes are attached to the whales, which record data such as location and interaction with other whales. Connected nodes transfer this data between each other. Whenever data is transferred to a base station (the paper proposes using seabirds), it can be collected and stored.

Because data is shared between nodes, it is no longer necessary to find a whale with a sensor in order to acquire data from that sensor. This is a perfect example of the power of opportunistic networks in an environment with very limited connectivity.

## 3.3 ROUTING ALGORITHMS

Routing messages in opportunistic networks is a non-trivial task because it is impossible to predict connections with any certainty.

Opportunistic networks can be viewed as a constantly changing graph. For this reason, many opportunistic routing algorithms are similar to graph search techniques. However, because the graph is constantly changing and is not necessarily random, such techniques are not necessarily the most effective (as shown by the Haggle project).

### 3.3.1 Context Based Routing

Context based routing is a form of greedy best-first search, where a single message is continually passed to the node most likely to reach the destination. There are a variety of methods to compute the utility of a node, including CAR (4) and MobySpace (5). While it is not guaranteed to find the optimum path (or any path) to the destination, it uses very few resources as the message is never copied.

### 3.3.2 Epidemic Routing

The opposite of context based routing is epidemic routing - a form of uniform cost search (6). Copies of the message are passed at every opportunity until the network is saturated. This is often likened to the spread of a virus. While this approach will always find the optimal path (because it takes all possible paths), it is very resource intensive - all nodes are expected to store every possible message. For this reason, routing protocols that use similar techniques (known as dissemination based routing) concentrate on reducing resource usage.

### 3.3.3 PROPHET

The Probabilistic Routing in Intermittently Connected Networks (PROPHET) algorithm (7) is related to the A* search algorithm. A utility function (derived from recent encounters with nodes) is used to predict whether a copy of the message should be passed on. This heuristic based approach uses fewer resources than traditional epidemic routing.

### 3.3.4 Bubble RAP

The Haggle project discovered that algorithms that treat routing as a generic graph search problem are often unsuited to PSNs. Bubble RAP (8) works on the idea that a social connections graph has tree like structure, where closely related nodes form a community. In order to send messages to a different community, the message is sent towards the highly connected nodes near the root, and then towards the destination community and, eventually, the destination node.

This has been shown using the data collected from Haggle to be much more effective than standard routing algorithms for sending messages to a known recipient (8).

## 3.4 SECURITY

Security can be compromised in an opportunistic network by controlling a node or by intercepting messages during transmission. Common attack types include:

- Sybil attacks: impersonating another node in order to send messages that appear to be from that node or to receive messages intended for the node.
- Majority attack: by controlling a large number of nodes, an attacker can control a network which assumes that the majority of nodes can be trusted.
- Eavesdropping: gathering information such as message metadata to discover private information such as message contents and user location.
- Denial of Service: saturating the network with unwanted messages.
- Black hole attack: failing to pass on messages to either reduce resource usage or as part of another attack.

### 3.4.1 Trust Based Security

Trust based security mechanisms depend on generating a list of trusted or untrusted nodes. This is commonly based on trusting connections within a social network (9) or distrusting nodes exhibiting strange behaviour (10).

### 3.4.2 Cryptographic Security

Conventional cryptographic security mechanisms often use a single trusted authority to verify identities and distribute certificates. This is infeasible in a scalable opportunistic network because as the network grows, the time to communicate with the central server increases. Some mechanisms, like the one proposed by Shikfa et al (11) do use a central server, but only

require it to be available for nodes joining the network. Other mechanisms split the responsibility over a number of nodes. Mechanisms for distributed certificate distribution require some level of trust in network nodes. For example Capkun et al's approach (12) does this by building a graph of certificates determining who trusts who - any abnormalities in the trust graph may indicate foul play.

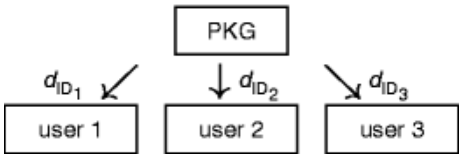
### 3.4.3 Identity Based Encryption

Identity based encryption is an increasingly common form of encryption where a user's unique identity (such as an email address) acts as a public key, and a secret key is generated by a central private key generator (PKG). The advantage of this approach is that public keys (IDs) are easily distributed and the central server only has to be contacted once (to fetch the secret key).

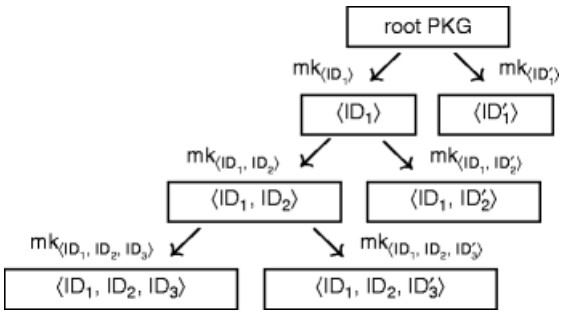
When applied to opportunistic networks, this approach has similar problems to certificate based approaches - a central server is needed. Some security frameworks assume that there is a central PKG that can and will be accessed occasionally (13). Others split up the PKG into multiple nodes, some of whom must collaborate to generate a secret key (14). The advantage of the identity based approach is that it is no longer necessary to distribute public keys - secret key distribution is still necessary but can happen less frequently (e.g. when a central PKG on the internet is available).

### 3.4.4 Hierarchical Identity Based Encryption

Hierarchical Identity Based Encryption (HIBE) is a form of IBE where any node with a secret key can generate a secret key for another node. For example, the central PKG generates a secret key for a user ID A. User A can now delegate a secret key for users A/B, A/C etc. This creates a tree hierarchy where the central PKG is the root and all other node's IDs describe the path to the root. Van Tilborg & Jajodia provide the following diagrams to explain the difference (15):




ID Based Encryption




Hierarchical ID Based Encryption

There have been no known applications of HIBE to opportunistic networking, although Seth & Keshav present a working solution for delay tolerant networks (16) which could be adapted for opportunistic networks.

# 4 USE CASES

The following use cases describe circumstances where conventional networks are infeasible, but opportunistic networks could still be used.

## 4.1 DISASTER AREA

A tsunami has wiped out all communications infrastructure in the area and injured a lot of people. Our opportunistic network is the quickest way contact medical teams to inform them of injured people who need help. We can assume that most people have smartphones and will be moving about regularly. Alice is injured and must contact the nearest free medical team so that they can help her. She uses her smartphone to publish a message with her location, and her status to all nearby nodes. The message is distributed in this manner until it reaches doctor Bob. Bob sends a reply message to indicate that help is coming and goes to help Alice.

## 4.2 PRIVACY

A group of activists are concerned that their internet access is being monitored by their government and their online identity could be hacked or blocked in order to oppress them. They use our network to communicate in a way that cannot be blocked or subverted while revealing as little metadata (such as location) as possible. In order to organise a protest, Alex, a well-known activist whose identity is unknown, sends out a message with details of the protest. Brian cannot come at that time because he has an exam, so he informs Alex of this via a secure direct message. Alex sends out a new message with a new date for the protest.

# 5 THREATS

From the use cases, we can construct a model of potential motives for attack and attack vectors.

## 5.1 DISASTER AREA

In a disaster area, people tend to act altruistically. Therefore they are unlikely to attempt to subvert the network. However, people may act selfishly by attempt to conserve battery or memory space on their phone.

- Goal: Selfishly reduce personal resource usage
  - Avoid accepting messages or passing them on (black hole attack)

## 5.2 PRIVACY

In this scenario, it is very important that the network cannot be compromised by attackers. These attackers may have significant resources available to them.

- Goal: Prevent a message from being disseminated
  - Force all nodes to discard message
    - Control most network nodes (majority attack)
      - Create lots of black hole nodes
      - Gain control of nodes
    - Overload most nodes (denial of service attack)
- Goal: Modify message while in transit
  - Control a node in the message path
    - Create node and get it into the message path
    - Gain control of a node in the path
- Goal: Prevent a user from sending messages
  - Force all nodes to discard messages from user
    - Control most network nodes (majority attack)
      - Create lots of black hole nodes
      - Gain control of nodes
    - Discredit user
      - Send fake messages from user (Sybil attack)
      - Exploit trust mechanism
  - Hack user's device
  - Legal action
    - Discover user identity
- Goal: Identify physical identity of a user

  - Derive network topology and route taken by a message to identify user location
  - Identify unique information about a user (such as phone number)
- Goal: Identify contents and recipient of an encrypted message
  - Derive route taken by a message to identify recipient
  - Discover private encryption key of recipient
    - Control master PKG
    - Brute force attack

## 5.3 ALL THREATS

Considering these scenarios and threats, the network must protect against the following attacks:

1. Sybil (impersonating another user)
2. Message modification
3. Majority (circumventing trust-based mechanisms by controlling the majority of the network)
4. Black hole (failing to pass on messages)
5. Denial of Service (overloading the network with messages)
6. Snooping (using metadata to infer private information)

# 6 OBJECTIVES

In order to implement an opportunistic network for the use cases, the following objectives should be met:

## 6.1 PRIMARY OBJECTIVES

- Design and implement a protocol for discovering nodes in close proximity and passing messages and necessary metadata between them.
- Create a core library to manage message storage and routing.
- Implement a simple epidemic routing algorithm to send messages to all available nodes.
- Design a routing algorithm using user metadata to route messages while disguising message content and metadata.

## 6.2 SECONDARY OBJECTIVES

- Create a smartphone UI.
- Implement a more advanced routing algorithm.
- Design and implement a mechanism to decide whether a node is trustworthy or not.
- Evaluate the performance of the implemented routing algorithms.

## 6.3 TERTIARY OBJECTIVES

- Compare the real world vs. simulated performance of the routing algorithms.

# 7 REQUIREMENTS SPECIFICATION

From the objectives and use cases, I have formulated a set of requirements which the system should meet in order to fulfil the objectives and be useful in the given use cases.

## 7.1 USER REQUIREMENTS

### *7.1.1 Non-Functional Requirements*

- High: The user shall be able to create a unique identity.
- High: The user shall be able to send plain text messages to all others who follow the user or a hashtag in the message.
- High: The user shall be able to 'follow' any user and receive messages sent by that user.
- Medium: The user shall be able to cancel any message that they have sent.
- Low: The user shall be able to 'follow' any hashtag and receive messages containing that hashtag.
- Low: The user shall be able to send encrypted direct messages to a single user.
- Low: The user shall be able to send multimedia messages in addition to plain text.

### *7.1.2 Functional Requirements*

- High: The user shall be able to use the network on their smartphone.
- Medium: The user shall be able to be confident in the origin and integrity of a message.
- Medium: The user shall be able to use the network with minimal training.

## 7.2 SYSTEM REQUIREMENTS

### *7.2.1 Non-Functional Requirements*

- High: The system shall work on smartphones or tablets capable of connecting to a wifi network.
- High: The system shall allow creation of user identities with a unique cryptographic identity.
- High: The system shall automatically connect to nearby nodes and pass on relevant information.
- Medium: The system shall provide a mechanism for securely distributing the cryptographic identity of a user.
- Medium: The system shall protect user metadata such as location and friends list from all other nodes.
- Medium: The system shall ensure that messages cannot be modified in transit or that such modifications can be detected.

- Medium: The system shall ensure that nodes cannot send a message that appears to be from another user.
- Medium: The system shall be robust and able to continue functioning when it encounters an unexpected state such as a malfunctioning or untrustworthy node.
- Medium: The system shall ensure that encrypted direct messages cannot be read by third parties.
- Low: The system shall be able to support multiple user identities on a single node.

### *7.2.2 Functional Requirements*

- High: The system shall collect anonymous logging data for debugging and profiling purposes.
- Medium: The system shall ensure that a reasonable number of messages reach their intended recipients.
- Medium: The system shall restrict the size of the message buffer by evicting messages.
- Medium: The system shall reduce power usage where possible.

# 8 SOFTWARE ENGINEERING PROCESS

## 8.1 TASK MANAGEMENT

The project was divided into a number of tasks to be completed. I recorded these in a spreadsheet along with their importance, timescale and dependencies (what tasks need to be completed beforehand). An example of the spreadsheet during the software development process is below:

| Number | Done | Status | Description | Type | Importance | Timescale |
|---|---|---|---|---|---|---|
| 17 | Y | Done | implement a simple epidemic routing algorithm to send message to all available nodes | objective | Primary | Days |
| 21 | Y | Done | Design encryption mechanism | task | Primary | Days |
| 20 | Y | Done | Design a routing algorithm using user metadata to route messages while disguising message content a | objective | Primary | Days |
| 44 | | Ready | Poster | objective | Primary | Days |
| 24 | | Ready | Design smartphone UI | task | Secondary | Days |
| 23 | | Blocked | Create a smartphone UI | objective | Secondary | Weeks |
| 28 | Y | Done | design trust mechanism | task | Secondary | Days |
| 26 | | Ready | Implement a more advanced routing algorithm | objective | Secondary | Weeks |
| 27 | | Blocked | Implement a mechanism to decide whether a node is trustworthy or not | objective | Secondary | Weeks |
| 31 | | Ready | Find users to participate | task | Secondary | Days |
| 32 | Y | Done | send logs to a central server | task | Secondary | Days |
| 33 | | Blocked | obtain performance data | task | Secondary | Weeks |
| 30 | | Blocked | Evaluate the performance of the implemented routing algorithms | objective | Secondary | Weeks |
| 36 | | Ready | set up simulation | task | Tertiary | Weeks |
| 37 | | Blocked | run simulation | task | Tertiary | Days |
| 35 | | Blocked | Compare the real world vs simulated performance of the routing algorithms | objective | Tertiary | Weeks |
| 42 | | Blocked | publish on app store | task | Tertiary | Days |
| 43 | Y | Done | publish on github | task | Tertiary | Days |

| | |
|---|---|
| Primary | 95% |
| Secondary | 22% |
| Tertiary | 20% |
| Total | 64% |

I practiced an iterative form of development where I first built a very simple system to detect nearby Bluetooth users and slowly added features from the task list to meet the requirements. While I had no continuous integration structure, I made sure that committed code would always compile and run.

## 8.2 VERSION CONTROL

The project was stored in the school's Mercurial repository at http://ndw.hg.cs.st-andrews.ac.uk/sh-proj. I also kept a backup on my personal GitHub account at http://github.com/neilw4/OMiN (the git-remote-hg plugin at https://github.com/felipec/git-remote-hg allows pushing to both Git and Mercurial repositories).

# 9 DESIGN

I spent a lot of time at the start of the project researching current technologies and carefully designing the routing algorithms and security systems to fulfil the requirements.

## 9.1 ROUTING

A routing algorithm takes all available information about a message and uses that information to decide where to send it. In trust based networks, this information is freely available to all trusted nodes (recipients, previous paths etc.). However, we have opted for a model where all nodes are considered untrustworthy. This means that we must obscure or remove all of this information, while still allowing a routing algorithm to use it.

Our network will be sending messages to multiple users, so dissemination based routing protocol is more useful. Algorithms like Bubble RAP have been shown to be very effective for pocket switched networks (8), but they rely on knowing the destination of the message.

Given the disadvantages of epidemic routing, we will use a variation of the PROPHET routing algorithm (7) to distribute messages. We can use a variation of the SSNR-OSNR algorithm (17) to obfuscate sensitive metadata.

In future, this could be combined with a variant of Bubble RAP to favour sending messages through highly connected nodes.

## 9.2 MESSAGE BUFFER EVICTION

When the message buffer is too large, it must evict a message. Ideally, this message will already be close to the destination. Nodes cannot know this information, but they can use heuristics to infer it - messages that have been forwarded to many nodes are likely to be widely distributed throughout the network and are therefore closer to the destination than the current node is. Therefore, nodes should evict the message that has been forwarded the most.

When a message is evicted, we must ensure that it is not received again - this could result in loops where a message is forwarded, evicted re-received and re-forwarded. We should use a bloom filter - a small fixed size data structure representing a set which can tell if an object is probably in the set or definitely not in the set (18). When a message is seen, it should be added to the bloom filter. Messages should only be accepted if they are not in the bloom filter - they have definitely not been seen.

## 9.3 ENSURING MESSAGE INTEGRITY

Steps must be taken to prevent Sybil attacks (impersonation of users), message modification and majority attacks. Many network protocols (8) (9) use heuristic algorithms to determine which nodes in a network to trust. I have decided against this approach because such it cannot

guarantee security, limits the number of useable nodes in a network and is often susceptible to a majority attack.

I have chosen to take a cryptographic approach where users use an asymmetric key pair to sign messages and verify their origin and integrity. This has the additional benefit that, with some cryptographic algorithms, we can encrypt a message for user X with X's public key, so that it can only be decrypted by X. This means that we can verify the origin of a user and the integrity of a message, which cannot be affected if the majority of the network is controlled by an attacker.

This cryptographic approach doesn't solve all of our problems, however: if we receive a message from user X, we must know X's public key in order to verify the message's origin. Most solutions to this problem use a trust-based approach to distributing public keys (12). However this approach is susceptible to majority attacks in the same way that any other trust-based scheme is. My solution is to use ID-based cryptography - public keys are now short, memorable IDs (usernames or email addresses) which are already known or, if they are not known, are easy to distribute (unlike conventional large keys, they can fit on a QR code or be passed on by word of mouth). The disadvantage of ID based encryption is that secret keys must be generated and distributed by a central PKG. There are a number of solutions to this problem:

- Seth & Keshav (16) use USB drives to distribute one-time symmetric keys which are used to communicate with the PKG over the network. However their solution is aimed at delay tolerant networks where round trip times are more reasonable.
- Kong et al (14) propose using multiple PKGs where one or more PKG must collaborate to generate a secret key. This removes the bottleneck of a central server, but requires more PKGs to be created and managed as the network scales.
- The simplest solution, taken by Kamat et al (13) is to assume that every node can directly access the central PKG via the internet when they create an ID.

I propose using a version of Kamat et al's scheme (13) with a modification to allow the case where the PKG is not accessible. I us a HIBE scheme where every node with a secret key is capable of becoming a PKG and issuing secret keys to other users.

If user A cannot access the PKG, they can still be authenticated by user B (giving them the identity B/A). User B is either authenticated by the PKG or another user, so there will always be a chain back to the PKG. If the master PKG isn't available via the internet, another node can act as a delegate PKG. In this way we can create a chain of key generators where the master PKG (accessible via the internet) delegates PKG responsibilities down the chain. A node's secret key will be compromised if one of its parents or ancestors is compromised, so it is wise to keep this chain as short as possible.

This disadvantage of this scheme is that it relies on trusting parents and ancestors - they are, by definition, capable of deriving their descendant's secret keys. We can increase the security of this scheme by allowing users to assume multiple identities: for example, if user A signs messages with secret keys B/A and C/A (i.e. receives a secret key from both parents B and C), both B and C must collaborate to derive all of A's secret keys. This has the added advantage that we can calculate the probability of a node's secret key being compromised, given the average probability that a node has been compromised.

## 9.4 ALTERNATIVE TO HIBE-BASED APPROACHES

In practice (see the implementation section), there is no HIBE implementation capable of signing messages (although such a scheme is presented in theory by Yuen & Wei (19). We can still use Kamat et al's approach (13) (a central PKG accessed over the internet), but we need to deal with the case where the nodes cannot access the PKG to obtain their secret key. We can allow users to send unsigned messages, but we have no fool proof way of determining the message's origin and authenticity - any node between the sender and receiver could maliciously modify the message.

To reduce the possibility of this happening, a node with a secret key can sign the message on behalf of the sender, guaranteeing that it cannot be modified for the rest of its journey to the sender.

It is possible to encounter multiple copies of an unsecured message that have been signed by different nodes. Since both copies are identical, it does not matter which version should be passed on. We should always choose the message signed by the lowest username alphabetically because this will reduce further instances of this problem later on (as the message will eventually converge towards the version signed by the lowest username).

## 9.5 PREVENTING BLACK HOLE ATTACKS

A black hole attack is where a node fails to store or pass on a message. This can be done for selfish reasons (to reduce storage usage) or to prevent a message from being distributed (this often requires a lot of collaborating nodes). Schemes such as IRONMAN (10) and RADON (20) store metadata about recent connections in order to find nodes which are failing to pass on connections and decrease their reputation (for example; A sends a message through B then B connects to C but doesn't forward the message. When A later connects to C they can figure out that B is a black hole). Disreputable nodes will not be sent new messages, effectively isolating them from the network.

OMiN uses a dissemination based routing algorithm where many copies of the message are spread through the network. While black hole attacks are a serious threat to context based routing (a single black hole can stop a message), it is a less significant threat in our network - many black holes are needed to prevent a message being disseminated. For this reason, protection against black holes is a low priority in the network and has not been implemented.

If it were to be implemented, an algorithm similar to IRONMAN or RADON would be used to detect and punish black hole nodes.

## 9.6 PREVENTING SNOOPING

Snooping is the use of metadata (like location) to infer private information (like a user's identity). The routing algorithm has been designed to use very little metadata - any metadata that is used is disguised in a bloom filter using the SSNR-OSNR algorithm (17).

## 9.7 PROTECTING THE PKG

The PKG is the only party which must be trusted by all nodes. If it is compromised then the attackers could gain access to the master secret key, which could be used to generate secret keys for all users and compromise the whole network.

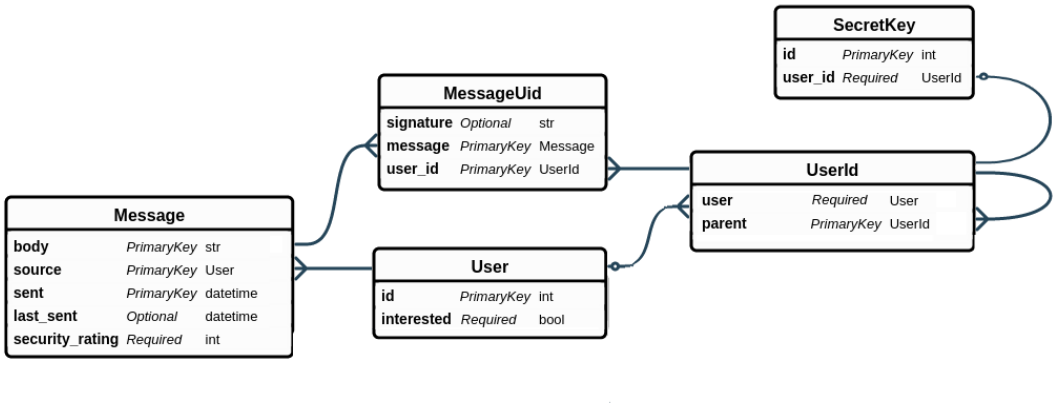
For this reason, the PKG must be built securely. It must be hosted on a secure system, transfer secret keys securely (using SSL) and be secured against injection attacks and unexpected input.

## 9.8 DATABASE DESIGN

Every node must store a record of messages and users encountered. The network can be seen as a way to synchronise these records – when a node is encountered, messages and other relevant metadata are taken from the database, converted to the JSON format and passed to the node, which deserialises the relevant messages and metadata and stores it in its database. This process encompasses the majority of the work of the network nodes.

Nodes store whether they are interested in receiving messages from a **user**. Every user can have multiple **user IDs**, which consist of a username and possibly a parent **user ID**. The user of the node also stores a **secret key** for every **user ID**. A **message** consists of a message body, source **user**, time when it was sent from the source, time when it was last sent by the node, a security rating (verified, unverified, unsigned) and one or more source **user ID**s, each with a signature.

The following ER diagram describes the database:


SecretKey
id PrimaryKey int
user_id Required UserId
MessageUid
signature Optional str
message PrimaryKey Message
user_id PrimaryKey UserId
UserId
user Required User
parent PrimaryKey UserId
Message
body PrimaryKey str
source PrimaryKey User
sent PrimaryKey datetime
last_sent Optional datetime
security_rating Required int
User
id PrimaryKey int
interested Required bool

# 10 IMPLEMENTATION

While implementing the network, I had to make a number of decisions about which technologies to use.

## 10.1 MOBILE PLATFORM

The network will work better as more people use it and more connections are formed. For this reason, it makes sense to use the most common platforms – Android and/or iOS. I chose to target Android devices because I have experience working with Android and no experience with iOS.

## 10.2 PROGRAMMING LANGUAGE

Android apps primarily use Java, but it is theoretically possible to use any language. However, only JVM languages (and C++ via the NDK) have access to the Android application framework and libraries. Scala is a very flexible JVM language I have used to write android apps before. The Scala build tools and libraries caused me a number of issues, so after some experimentation I started using Java because it is so well supported.

## 10.3 ENCRYPTION SCHEME

### *10.3.1Requirements*

The encryption algorithm should implement the following:

- Public keys are small enough to be distributed easily.
- Users can start without having to contact a central server to obtain a secret key.
- ID hierarchy should be unbounded (unrestricted in depth) - i.e. PKG authenticates who authenticates and so on up to for some arbitrary .
- Verifiable message source.
- Verifiable message integrity.
- Message contents can be obscured from all but the intended recipient.
- Encryption scheme will not be broken in the foreseeable future.
- Encryption scheme has an existing implementation which will work on the target platform (Android).

### *10.3.2Disaster Area Scenario*

In the disaster scenario, the following requirements are necessary:

- Small public keys.
- Users can start without having to contact a central server.
- Verifiable message integrity.

### 10.3.3Privacy Scenario

- In the privacy scenario, the following requirements are necessary:
- Verifiable source.
- Verifiable integrity.
- Unbroken encryption scheme.
- Obscured message contents.

All other requirements are optional but would improve the flexibility of the network.

### 10.3.4Algorithm Choice

I considered a number of encryption schemes – the most applicable schemes are listed below:

| | Paper | Scheme | Signing | Encryption | Implementation |
|---|---|---|---|---|---|
| LW11 | Lewko & Waters (21) | Unbounded HIBE | No | Yes | http://gas.dia.unisa.it/projects/jpbc/schemes/uhibe_lw11.html |
| DIP10 | De Caro, Iovino & Persiano (22) | Bounded HIBE | No | Yes | http://gas.dia.unisa.it/projects/jpbc/schemes/ahibe_dip10.html |
| PS06 | Paterson & Schuldt (23) | IBE | Yes | No | http://gas.dia.unisa.it/projects/jpbc/schemes/ibs_ps06.html |

LW11 is more suitable than DIP10 because it does not impose restrictions on the depth of the hierarchy.

The ideal algorithm would be an unbounded HIBE supporting signing and encryption, but no public implementations of such an algorithm exist. There appear to be no public implementation of any HIBE for signing either. Given that we must sign messages to verify their origin, the network implementation must use an IBE scheme. The best choice is a combination of PS06 for signing messages and LW11 for encrypting them where nodes store two secret keys – a PS06 key for signing and an LW11 key for decryption. Section 8.4 discusses a new design for the cryptography system taking into account that we can only use an IBE scheme for message signing.

## 10.4 DATABASE LIBRARY

The app needs to store messages and other data in a database. In order to simplify implementation, I decided to use an Object Relational Model (ORM) library to allow database records to be treated as objects. Some research showed that the Sugar ORM (http://satyan.github.io/sugar) library provided the necessary functionality and was easy to integrate with the application.

## 10.5 MESSAGE PASSING MEDIUM

There are a number of methods for smartphones in close proximity to interact. I considered the following methods:

- LAN communication - easy to implement but requires a LAN, which may not be possible for many use cases.
- Wifi-Direct - good range but requires that smartphones are not connected to a LAN.
- Bluetooth Low Energy - only supported by Android API 18+ (about 25% of devices) and research has shown that it is only as efficient as normal Bluetooth **[CITATION NEEDED]**.
- Bluetooth - well supported although limited connectivity.

Considering the pros and cons of all of them, I decided to use Bluetooth to pass messages as it is almost universally supported and does not rely on smartphones being connected or disconnected from a LAN.

## 10.6 PKG SERVER

The PKG server is a single point of failure, so it has to be very secure. I trust the department to setup a secure system more than I trust myself, so I have built the PKG software to run on the school's server, dealing with HTTPS requests via CGI. The server is written in Java because some code (like the encryption/decryption code) must be shared between the server and client. Java isn't particularly suited to CGI because a new JVM has to be created for every request (a relatively slow and costly operation), but there is a description of how to do it at http://www.javaworld.com/article/2076863/java-web-development/write-cgi-programs-in-java.html. The server maintains a list of users with secret keys so it will only ever send the secret key for a user once. Unix file locks are used to make sure that this file is only being read/written by one process at a time.

# 11 ETHICS

In order to test the real world performance of the network, we may ask people to use the application. In this case, some metadata will be collected on users, with their consent. This may include:

- An anonymous user ID.
- Anonymised 'Friends list' (or equivalent) of users.
- Metadata of messages passed during encounters, including message ID and origin ID, but NOT message contents.
- Times and locations of encounters between anonymous users.

# 12 EVALUATION AND CRITICAL APPRAISAL

**TODO**

# 13 CONCLUSIONS

**TODO**

# 14 BIBLIOGRAPHY


1. Scott J, Crowcroft J, Hui P, Diot C, Others. Haggle: A networking architecture designed around mobile users. In: WONS 2006: Third Annual Conference on Wireless On-demand Network Systems and Services; 2006. p. 78-86..

2. Chaintreau A, Hui P, Crowcroft J, Diot C, Gass R, Scott J. Impact of human mobility on opportunistic forwarding algorithms. Mobile Computing, IEEE Transactions on. 2007;6(6):606-620..

3. Small T, Haas ZJ. [Online].; The Shared Wireless Infostation Model: A New Ad Hoc Networking Paradigm (or Where There is a Whale, There is a Way). In: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking [cited Computing. MobiHoc '03. New York, NY, USA: ACM; 2003. p. 233-244. Available from: http://doi.acm.org/10.1145/778415.778443.

4. Musolesi M, Hailes S, Mascolo C. [Online].; Adaptive routing for intermittently connected mobile ad hoc networks. In: World of Wireless Mobile and Multimedia Networks, 2005. WoWMoM 2005. Sixth IEEE International Symposium on a; 2005. p. 183-189.. Available from: http://dx.doi.org/10.1109/WOWMOM.2005.17.

5. Leguay J, Friedman T, Conan V. [Online].; Evaluating Mobility Pattern Space Routing for DTNs. In: INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings; 2006. p. 1-10. Available from: http://dx.doi.org/10.1109/INFOCOM.2006.299.

6. Vahdat A, Becker D. ; Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University; 2000.

7. Lindgren A, Doria A, Schelén O. [Online].; Probabilistic Routing in Intermittently Connected Networks. SIGMOBILE Mob Comput Commun Rev. 2003 Jul;7(3):19-20. Available from: http://doi.acm.org/10.1145/961268.961272.

8. Hui P, Crowcroft J, Yoneki E. [Online].; Bubble Rap: Social-based Forwarding in Delay Tolerant Networks. In: Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing. MobiHoc '08. New York, NY, USA: ACM; 2008. p. 241-250. Available from: http://doi.acm.org/10.1145/1374618.1374652.

9. Trifunovic S, Legendre F, Anastasiades C. [Online].; Social Trust in Opportunistic Networks. In: INFOCOM IEEE Conference on Computer Communications Workshops, 2010; 2010. p. 1-6. Available from: http://dx.doi.org/10.1109/INFCOMW.2010.5466696.

10. Bigwood G, Henderson T. [Online].; IRONMAN: Using Social Networks to Add Incentives and Reputation to Opportunistic Networks [cited In: Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third International Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on; 2011. p. 65-72. Available from: http://dx.doi.org/10.1109/PASSAT/SocialCom.2011.60.

11. Shikfa A, Onen M, Molva R. [Online].; Bootstrapping security associations in opportunistic networks. In: Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on; 2010. p. 147-152. Available from: http://dx.doi.org/10.1109/PERCOMW.2010.5470676.

12. Capkun S, Buttyan L, Hubaux JP. [Online].; Self-organized public-key management for mobile ad hoc networks. Mobile Computing, IEEE Transactions on. 2003 Jan;2(1):52-64. Available from: http://dx.doi.org/10.1109/TMC.2003.1195151.

13. Kamat P, Baliga A, Trappe W. [Online].; An Identity-based Security Framework For VANETs. In: Proceedings of the 3rd International Workshop on Vehicular Ad Hoc Networks. VANET '06. New York, NY, USA: ACM; 2006. p. 94-95. Available from: http://doi.acm.org/10.1145/1161064.1161083.

14. Kong J, Petros Z, Luo H, Lu S, Zhang L. [Online].; Providing robust and ubiquitous security support for mobile ad-hoc networks. In: Network Protocols, 2001. Ninth International Conference on; 2001. p. 251-260. Available from: http://dx.doi.org/10.1109/ICNP.2001.992905.

15. Van Tilborg, Henk CA, Jajodia, Sushil. [Online].; Encyclopedia of cryptography and security. Springer Science & Business Media; 2011. Available from: http://link.springer.com/referenceworkentry/10.1007%2F978-1-4419-5906-5_148/fulltext.html.

16. Seth A, Keshav S. [Online].; Practical security for disconnected nodes. In: Secure Network Protocols, 2005. (NPSec). 1st IEEE ICNP Workshop on; 2005. p. 31-36. Available from: http://dx.doi.org/10.1109/NPSEC.2005.1532050.

17. Parris I, Henderson T. [Online].; Privacy-enhanced social-network routing. Computer Communications. 2012;35(1):62-74. Available from: http://www.sciencedirect.com/science/article/pii/S0140366410004767.

18. BH, Bloom. [Online].; Space/Time Trade-offs in Hash Coding with Allowable Errors. Commun ACM. 1970 Jul;13(7):422-426. Available from: http://doi.acm.org/10.1145/362686.362692.

19. Yuen TH, Wei VK. [Online].; Constant-Size Hierarchical Identity-Based Signature/Signcryption without Random Oracles; 2005. Kwwei@ie.cuhk.edu.hk, thyuen4@ie.cuhk.edu.hk 13302 received 17 Nov 2005, last revised 2 Jun 2006. Available from: http://eprint.iacr.org/2005/412.

20. Li N, Das SK. [Online].; RADON: Reputation-assisted Data Forwarding in Opportunistic Networks. In: Proceedings of the Second International Workshop on Mobile Opportunistic Networking. MobiOpp '10. New York, NY, USA: ACM; 2010. p. 8-14. Available from: http://doi.acm.org/10.1145/1755743.1755746.

21. Lewko A, Waters B. [Online].; Unbounded HIBE and attribute-based encryption. In: Advances in Cryptology--EUROCRYPT 2011. Springer; 2011. p. 547-567. Available from: http://dx.doi.org/10.1007/978-3-642-20465-4_30.

22. De Caro A, Iovino V, Persiano G. [Online].; Fully secure anonymous hibe and secret-key anonymous ibe with short ciphertexts. In: Pairing-Based Cryptography-Pairing 2010. Springer; 2010. p. 347-366. Available from: http://dx.doi.org/10.1007/978-3-642-17455-1_22.

23. Paterson KG, Schuldt JCN. [Online].; Efficient identity-based signatures secure in the standard model. In: Information Security and Privacy. Springer; 2006. p. 207-222. Available from: http://dx.doi.org/10.1007/11780656_18.