

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.

MORE SPECIFIC ABOUT WHAT IT DOES

INCLUDE SUMMARY OF EVALUATION

REFERENCES IN SQUARE BRACKETS

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	Introduction5
2	User Manual	Error! Bookmark not defined.
2.1	Source Code.....	Error! Bookmark not defined.
2.2	Project Layout.....	Error! Bookmark not defined.
2.3	Building.....	Error! Bookmark not defined.
2.4	Installation.....	Error! Bookmark not defined.
2.5	Use.....	Error! Bookmark not defined.
3	Background.....	.6
3.1	Microblogging6
3.2	Opportunistic Networks6
3.3	Similar Projects.....	.7
3.4	Routing Algorithms9
3.5	Security12
4	Use Cases.....	.16
4.1	Disaster Area16
4.2	Metro16
5	Threats18
5.1	Disaster Area18
5.2	Metro18
5.3	All Threats.....	.19
6	Objectives20
6.1	Primary Objectives.....	.20
6.2	Secondary Objectives20
6.3	Tertiary Objectives20
7	Requirements Specification21
7.1	User Requirements.....	.21
7.2	System Requirements21
8	Software Engineering Process23
8.1	Task Management.....	.23
8.2	Implementation23
8.3	Storage.....	.23
9	Design.....	.24
9.1	Social Structure24
9.2	Routing.....	.24
9.3	Message Buffer Eviction25
9.4	Ensuring Message Integrity.....	.25
9.5	Alternative to HIBE-Based Approaches.....	.26
9.6	Preventing Black Hole Attacks27
9.7	Preventing Snooping.....	.28
9.8	Protecting the PKG.....	.28

9.9 Database Design.....	28
10 Implementation	29
10.1 Mobile Platform	29
10.2 Programming Language.....	29
10.3 Message Passing Medium	29
10.4 Database Library	30
10.5 Encryption Scheme.....	30
10.6 PKG Server.....	31
11 Ethics.....	33
12 Evaluation and Critical Appraisal.....	34
12.1 Testing.....	34
12.2 Objectives	34
12.3 Requirements	34
12.4 Use Cases.....	34
12.5 Further Work.....	34
12.6 Overall Evaluation	35
13 Conclusions.....	36
14 Bibliography.....	37
15 Appendices.....	38
15.1 Appendix 1 - User Manual	39

1 INTRODUCTION

TODO

2 BACKGROUND

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.

2.1 MICROBLOGGING

REFERENCES – SOCIAL NETWORK AND MICROBLOGS

WHAT IS A SOCIAL NETWORK

REFERENCE - DANAH BOYD SOCIAL NETWORK SITES

A microblog is a simple form of social network where users post short messages which can be viewed by others. Most social networks use some form of microblogging – Twitter's 140 character tweets and Facebook's short status updates.

EXPAND

2.2 OPPORTUNISTIC NETWORKS

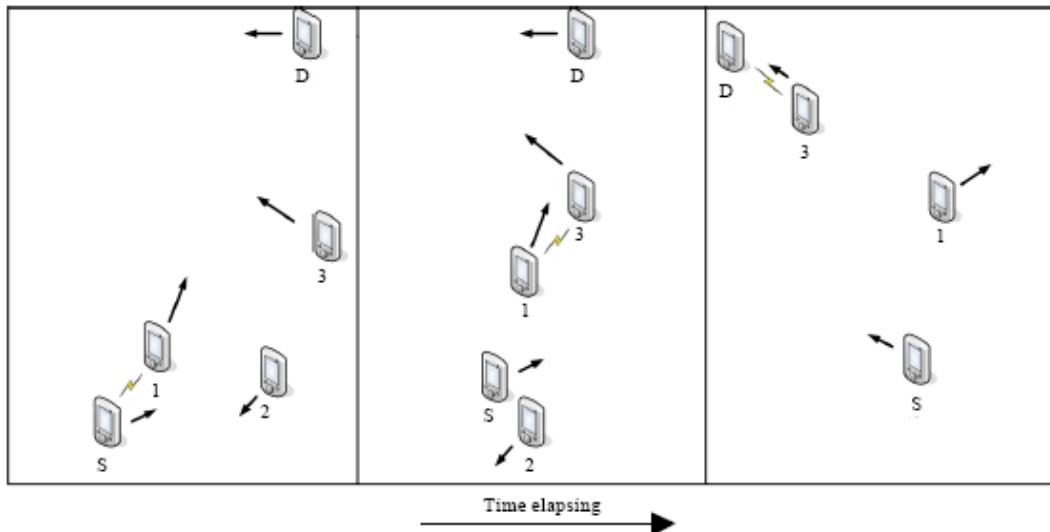
REFERENCES

WHAT PROBLEMS DOES IT SOLVE? – WHY USE 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.

The following diagram shows how a message might be sent from sender S to destination D in a Pocket Switched Network. S encounters node 1. The routing algorithm calculates that the message is more likely to reach D if it is sent through node 1, so node 1 receives the message. Later on, node 2 encounters node 3 and passes on the message because node 3 is likely to reach the destination. Node 3 eventually meets D and passes on the message.



BIGGER PICTURES

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.

2.3 SIMILAR PROJECTS

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

2.3.1 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 (<http://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).

2.3.2 *FireChat*

FireChat (<http://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>).

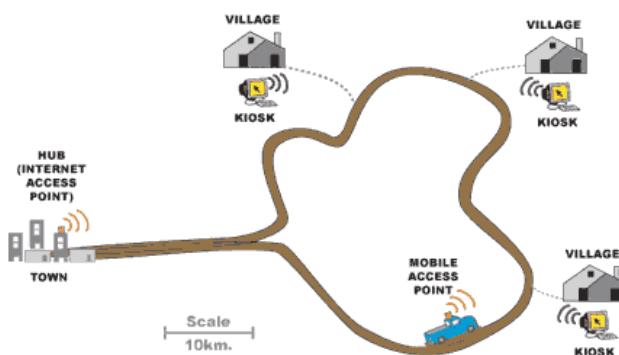
However, the app mostly relies on an internet connection, and its simple protocol is insecure (http://breizh-entropy.org/~nameless/random/posts/firechat_and_nearby_communication) and unable to implement the store-and-forward functionality of a proper opportunistic network.

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

FOOTNOTES FOR URLs

2.3.3 *Daknet*

Daknet (3) is an opportunistic network in rural villages in India. Each village has a kiosk which can connect wirelessly to Mobile Access Points (MAPs) on buses and cars. This access point travels between villages and towns, carrying communications between them. As well as carrying communications between villages, a MAP can also relay requests to download something from the internet from an internet access point.



2.3.4 *SWIM*

The Shared Wireless Infostation Model (SWIM) is a proposed opportunistic network to monitor whales (4). Small nodes are attached to the whales, which record data such as location and interactions 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.

SWIM makes it possible to get information about a whale without having to physically find it to read data from its sensor – data about the whale will have been relayed to sensors on other whales. This is a perfect example of the power of opportunistic networks in an environment with very limited connectivity.

2.4 ROUTING ALGORITHMS

START WITH EPIDEMIC

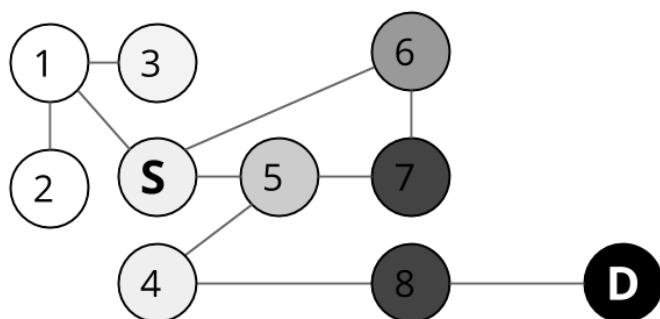
HOT POTATO VS DISSEMINATION SECTION

COMPARE ROUTING ALGORITHMS MORE

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

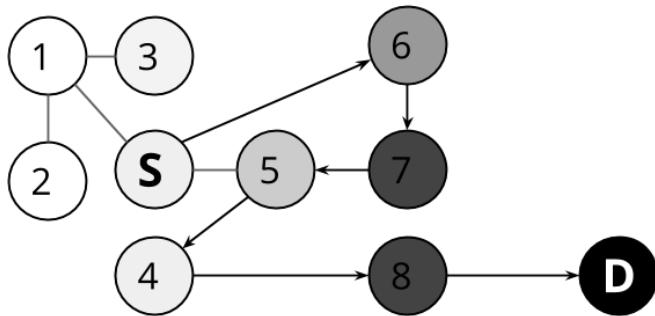
Opportunistic networks can be viewed as a constantly changing graph. For this reason, many opportunistic routing algorithms are similar to graph search techniques. However, because the graph is constantly changing and is not necessarily random, such techniques are not necessarily the most effective (as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

For demonstration purposes, we will use the following network which is trying to pass a message from source S to destination D. The colour of a node represents some heuristic measure of utility (distance to D), where darker nodes are closer to the destination and lighter nodes are further away. I have done my best to mirror the complex and unpredictable encounters in an opportunistic network.



2.4.1 Context Based Routing

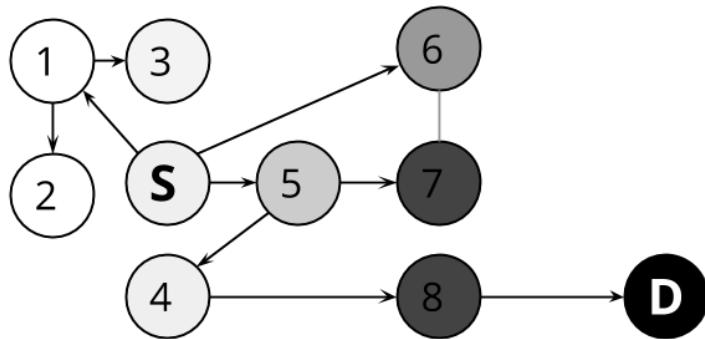
Context based routing is a form of greedy best-first search, where a single message is continually passed to the node most likely to reach the destination. There are a variety of methods to compute the utility of a node, including CAR (5) and MobySpace (6).



In the above example, a single copy of the message is always passed to the neighbouring node closest to the destination. 5 nodes are visited by the message (6-7-5-4-8), which is far from the optimal path (3 nodes – 5-4-8) but correctly avoids node 1. As with all heuristic algorithms, there are pathological cases – a naïve implementation would get stuck in an infinite loop (S-6-5-7-S-6...) if S was closer to D than 4. Similarly, if the only path to D was through node 1, the algorithm would never find it. 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, so is a good choice for networks with a good heuristic for node utility.

2.4.2 Epidemic Routing

The opposite of context based routing is epidemic routing - a form of uniform cost search (7). Copies of the message are passed at every opportunity until the network is saturated. This is often likened to the spread of a virus.

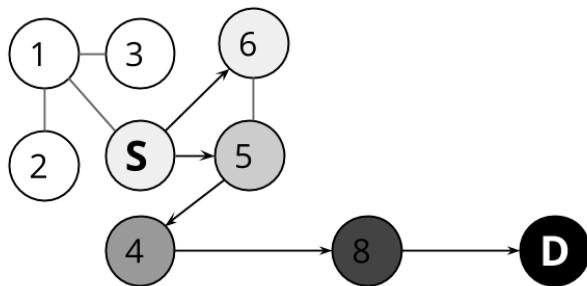


In the above example, the message is passed to all nodes, eventually reaching the destination by the shortest path. While this approach will find the optimal path (because it takes all possible paths), it is very resource intensive because all nodes are expected to store every possible message. If a node cannot store every possible message, the algorithm may not be able to find the optimal path. For this reason, routing protocols that use similar

techniques (known as dissemination based routing) concentrate on reducing resource usage.

2.4.3 PROPHET

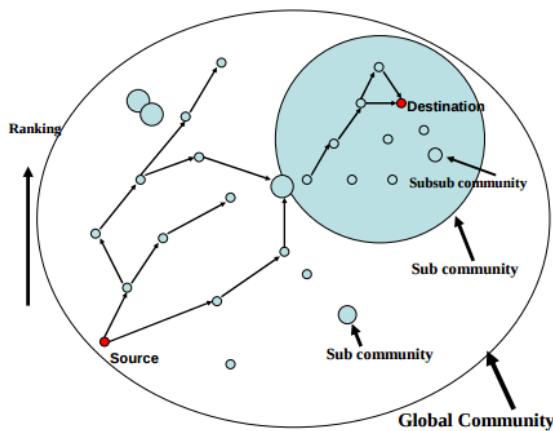
The Probabilistic Routing in Intermittently Connected Networks (PROPHET) algorithm (8) is a common dissemination-based algorithm. The algorithm relies on the heuristic that if a node was encountered recently, it is likely to be encountered again. A node's utility is derived from the utility of recently encountered nodes. Messages are only passed on to nodes with a higher utility than the sender. Messages follow a path which would recently have brought them to the destination.



The algorithm normally performs very well, often finding the best route and using fewer resources than naïve epidemic routing.

2.4.4 Bubble RAP

The Haggle project discovered that algorithms that treat routing as a generic graph search problem are often unsuited to PSNs. Bubble RAP (9) works on the idea that a social connections graph has a tree like structure, where closely related nodes form a community (a bubble) and highly connected, high ranking nodes near the root can forward messages between communities. In order to send messages to a different community, the message is sent towards the highly connected nodes with a high ranking, and then towards the destination community and sub-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 (9).

2.5 SECURITY

WHY IS THIS A PROBLEM VS CONVENTIONAL 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.

2.5.1 *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 (10) or distrusting nodes exhibiting strange behaviour (11).

2.5.2 *Cryptographic Security*

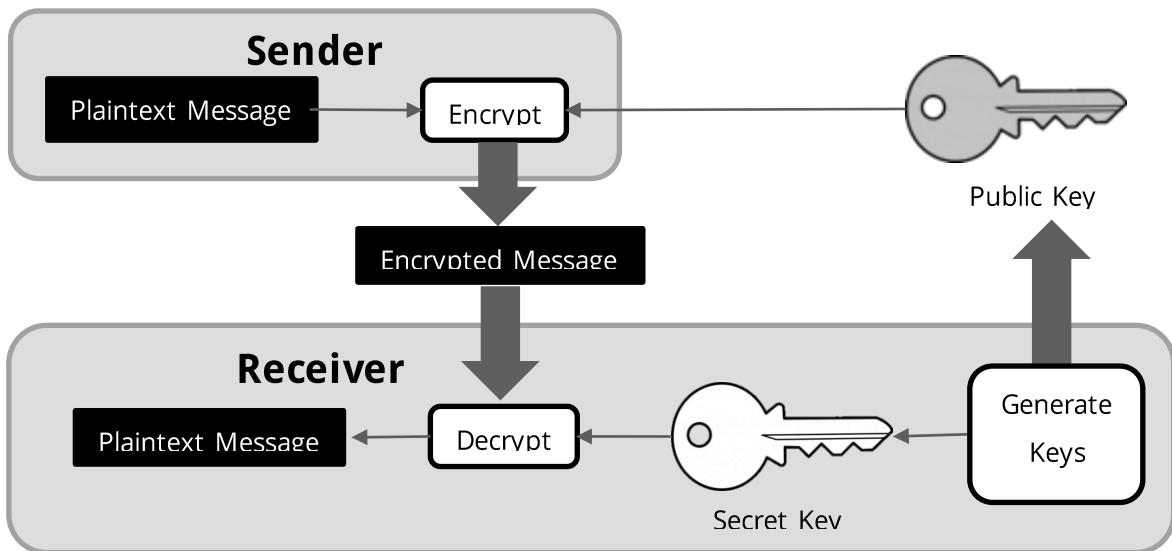
Conventional cryptographic security mechanisms often use a single trusted authority to verify the identity of a user and distribute certificates.

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 (12) 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 (13) does this by building a graph of certificates determining who trusts who - any abnormalities in the trust graph may indicate foul play.

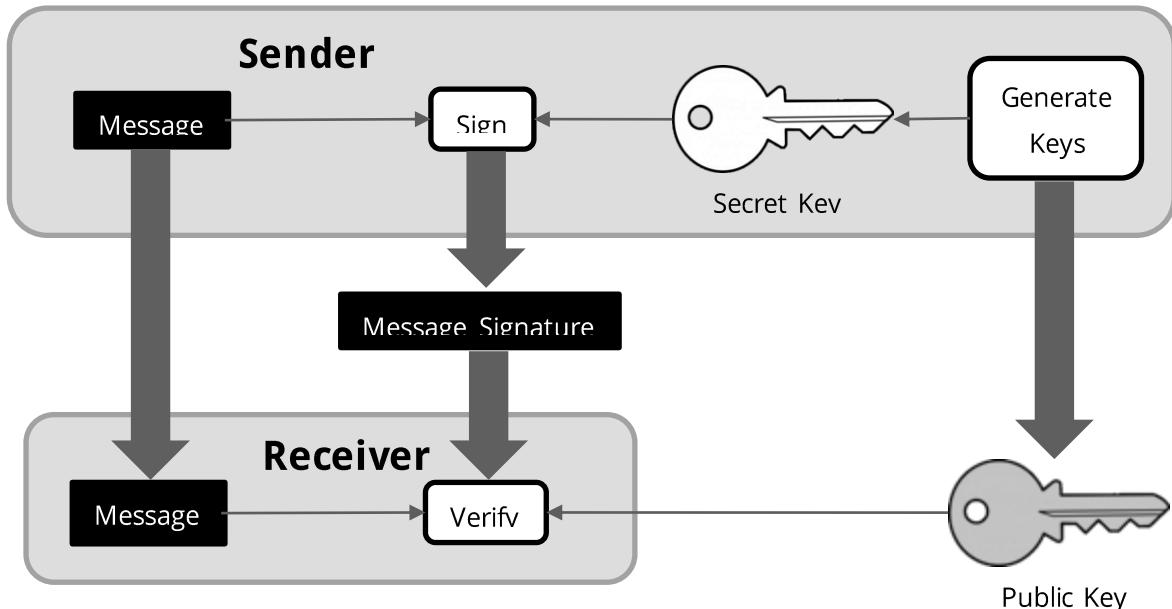
2.5.3 Asymmetric Key Cryptography

Asymmetric key cryptography is a commonly-used method for encrypting data and signing it to verify its origin and integrity. Every user has two cryptographic keys – a public key known to the world and a private secret key known only to them.

The public key can be used to encrypt a message so that it can only be decrypted by the matching secret key. Since only the intended recipient knows the secret key, only they can decrypt it.



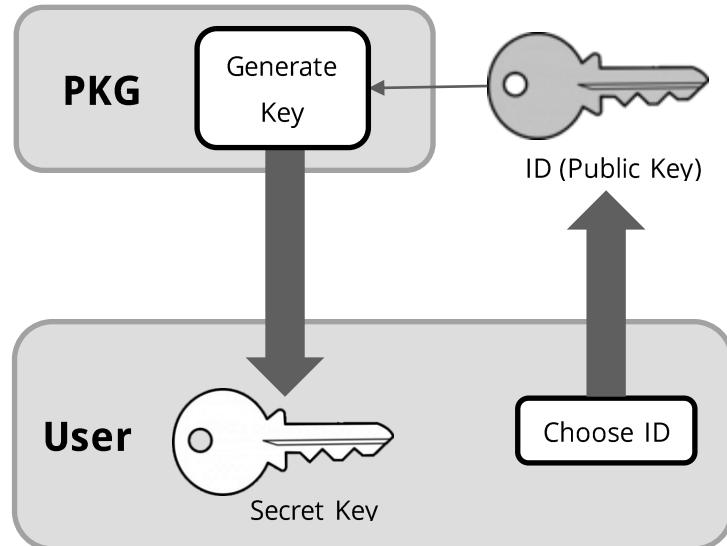
Similarly, the secret key can be used to create a message signature. This signature can be used to verify that the message creator knows the secret key and that the message has not been modified after it was created (data integrity).



2.5.4 Identity Based Encryption

REFERENCE

Identity based encryption is an increasingly common form of asymmetric key encryption where a user's public key is a short unique identifier (such as an email address), and a secret key is generated by a central private key generator (PKG) and sent securely to the user.

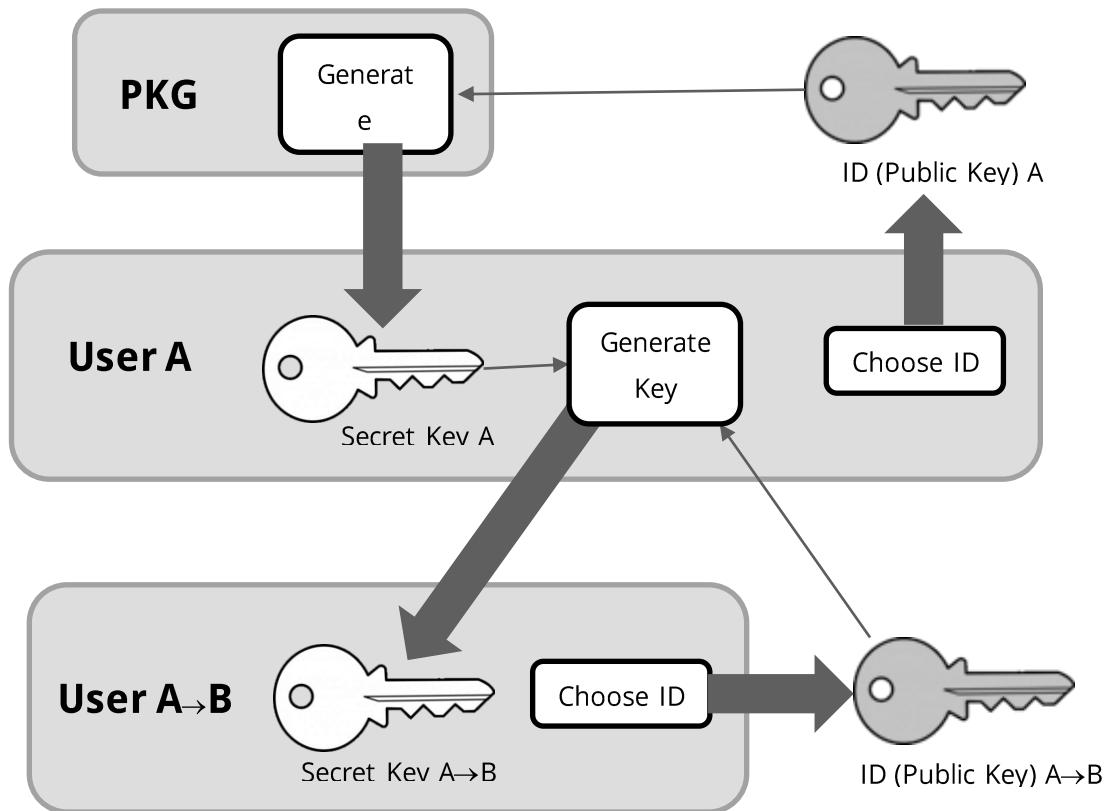


When applied to opportunistic networks, this approach has similar problems to certificate based approaches - a central server is needed. Some security frameworks assume that there is a central PKG that can and will be accessed occasionally (14). Others split up the PKG into multiple nodes, some of whom must collaborate to generate a secret key (15). The advantage of the identity based approach is that it is no longer necessary to distribute public keys – it is still necessary for the PKG to distribute secret keys, but this can happen less frequently (e.g. when a central PKG on the internet is available).

2.5.5 Hierarchical Identity Based Encryption

REFERENCE

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.



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.

2.5.6 SSNR-OSNR Obfuscation

TODO

(17)

3 USE CASES

THERE ARE LOTS OF USE CASES (CITE TRISTAN'S DTN PAPER?)

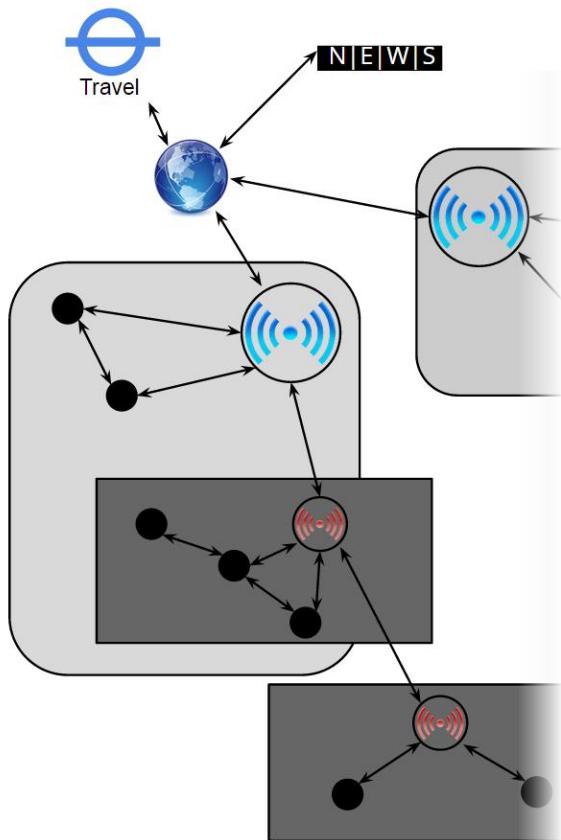
The OMiN network is designed as a general opportunistic network, focusing on the following scenarios where communication over conventional networks is infeasible.

3.1 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.

3.2 METRO

Many people use the London underground to commute to work. Because it is underground there is no mobile phone signal and no way of connecting to the internet. Commuters use the OMiN network as a social network where conventional internet based networks will not work. As well as messages from individual users, internet-connected base stations at subway stations send travel updates from Transport for London and breaking news from the BBC and other news sites to commuters and message distribution nodes on the trains. Commuters choose which people and updates they wish to receive messages from and can also view the most popular messages. Anonymous messages are also permitted. Users can also send encrypted direct messages to other users such that the message can only be decrypted and read by the other user. Messages from unpopular users will not spread far but I expect that a small number of users will become very popular for their amusing or novel messages – their messages will reach the whole network.



TODO: EXPLAIN PIC

4 THREATS

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

EXPLAIN THREAT TREES

4.1 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)

4.2 METRO

In this scenario, an attacker could disrupt the transport network by sending messages that appear to be travel updates from Transport for London or breaking news from news sites. It is very important to ensure that these popular users aren't compromised.

- Goal: Spread false information appearing to be from a reputable source (Sybil attack)
 - Exploit trust-based security mechanism (if it exists)
 - Discover secret key of reputable source
 - Hack source
 - Hack PKG
- Goal: discover contents of an encrypted direct message
 - Discover secret key of recipient.
 - Hack recipient
 - Hack PKG
- Goal: bring down whole network
 - Prevent all messages from being disseminated
 - Control most influential network nodes (majority attack) and turn them into black holes
 - Gain control of most message dissemination nodes in trains
 - Gain control of most base stations in subway stations
 - Flood the network with messages (Denial of Service)

4.3 ALL THREATS

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

1. Sybil (impersonating another user)
2. Message modification
3. Black hole (failing to pass on messages)
4. Denial of Service (overloading the network with messages)
5. Snooping (using metadata to infer private information)

5 OBJECTIVES

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

5.1 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.

5.2 SECONDARY OBJECTIVES

- Create a smartphone user interface.
- 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.

5.3 TERTIARY OBJECTIVES

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

6 REQUIREMENTS SPECIFICATION

From the objectives, use cases and threats, 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.

6.1 USER REQUIREMENTS

6.1.1 *Functional Requirements*

- High: The user shall be able to use the network on their smartphone.
- 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.
- High: The user shall be able to 'follow' any user and receive messages sent by that user.
- High: The user shall be able to send messages without requiring an internet connection.
- Medium: The user shall be able to send anonymous messages without their username.
- Medium: The user shall be able to 'follow' any hashtag and receive messages containing that hashtag.
- Low: The user shall be able to send messages via the internet to internet-connected nodes.
- Low: The user shall be able to send encrypted direct messages to a single user.

6.1.2 *Non-Functional Requirements*

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

6.2 SYSTEM REQUIREMENTS

6.2.1 *Functional Requirements*

- High: The system shall work on portable electronic devices such as smartphones or tablets.
- 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.
- High: The system shall pass on messages until they reach their destination.
- 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 restrict the size of the message store.
- Medium: The system shall protect against Sybil attacks.
- Medium: The system shall prevent messages from being modified while in transit.
- Medium: The system shall protect user metadata from all other nodes.
- Low: The system shall prevent attempts to prevent message propagation.
- Low: The system shall have mechanisms to mitigate Denial of Service attacks.

6.2.2 Non-Functional Requirements

- High: The system shall work in an unstructured environment with random encounters between nodes.
- High: The system shall not require a connection to any other network (such as the internet).
- 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 minimise the number of messages lost before they reach their destination.
- Medium: The system shall route messages effectively given a semi-predictable set of encounters between nodes.
- Medium: The system shall deliver messages as quickly as possible.

7 SOFTWARE ENGINEERING PROCESS

7.1 TASK MANAGEMENT

ACTIVE VOICE – I DID THIS

The project is divided into a number of tasks to be completed. These are recorded along with their importance, timescale and dependencies (what tasks need to be completed beforehand). Below is a snapshot of the tasks during the software development process.

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	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%					

7.2 IMPLEMENTATION

The network is built in an iterative manner, starting with a simple system to detect nearby Bluetooth users and adding features from the task list to move towards the objectives. Once a task is completed and the code is in a working state, it is committed to the repository.

7.3 STORAGE

The project is stored in the school's Mercurial repository at <http://ndw.hg.cs.st-andrews.ac.uk/sh-proj>, with 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).

8 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.

8.1 SOCIAL STRUCTURE

Both use cases revolve around users viewing short messages sent by others. For this reason, it seems practical to structure the network as a microblogging service similar to Twitter, where users send short messages and are “followed” by other users. Once this is operational, it is possible to add more advanced features like encrypted direct messages and “hashtags”.

8.2 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 I have opted for a more secure model where all nodes are considered untrustworthy – information must be hidden or obscured while still allowing a routing algorithm to use it.

Algorithms like Bubble RAP have been shown to be very effective for pocket switched networks (9) but they use context based routing where a message is sent to a single known destination. OMiN cannot use this approach because messages can be sent to multiple, possibly unknown, users. A dissemination based protocol like epidemic routing is much more useful in this environment.

The PROPHET routing algorithm (8) is a very good dissemination based routing protocol which can be adapted to minimise the use of sensitive metadata. In PROPHET, all nodes calculate and advertise the probability that they will be able to deliver the message to its destination. If node A has communicated with the destination recently, it has a high probability. When node B communicates with node A, it calculates that its probability is lower than A's.

DON'T USE CONTRACTIONS

OMiN messages are microblogs which don't have a specific unique destination – they are sent to all users expressing interest in messages from the sender. Instead of distinguishing between destination nodes and carrier nodes, all destination nodes can advertise that they have a 100% probability.

OMiN will use this strategy combined with 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 detect communities and increase performance.

8.3 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.

TODO: PIC

8.4 ENSURING MESSAGE INTEGRITY

Steps must be taken to prevent Sybil attacks (impersonation of users), message modification and majority attacks. Many network protocols (9) (10) 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 (13). 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 (15) 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 (14) 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 (14) 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.

TODO: PIC

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.

8.5 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 (14) (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.

INTERNET – CAPITAL I

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).

TODO: PIC

8.6 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 (11) 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.

TODO: PIC

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.

8.7 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).

8.8 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.

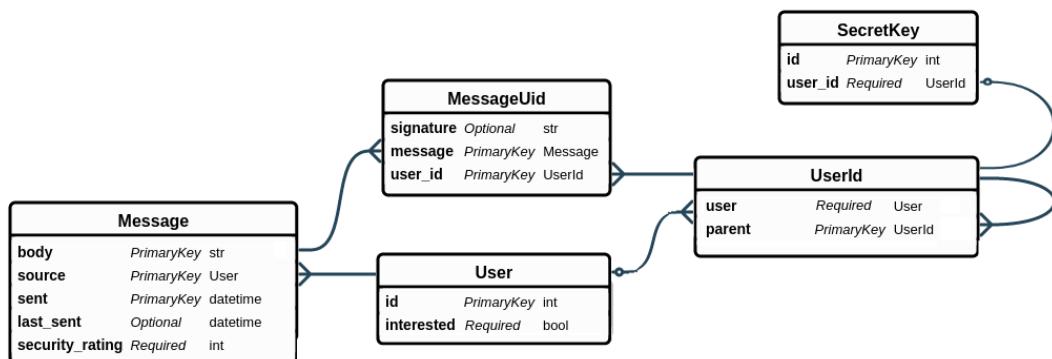
For this reason, the PKG must be built securely. It must be hosted on a secure system, transfer secret keys securely (using SSL) and be invulnerable to injection attacks and unexpected input.

8.9 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 IDs, each with a signature.

The following ER diagram describes the database:



9 IMPLEMENTATION

LINK TO REQUIREMENTS

There are a number of decisions to make about which technologies to use to implement the network.

TODO: REORDER?

9.1 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 will target Android devices because I have experience working with Android and no experience with iOS.

9.2 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, so I started writing the program in Scala. However, the build tools and libraries caused me a number of issues which forced me to switch back to Java because it is so well supported.

9.3 MESSAGE PASSING MEDIUM

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

- 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 - MARTIN].

- Bluetooth - well supported although limited connectivity.

The best option is to use Bluetooth because it is almost universally supported and does not rely on smartphones being connected or disconnected from a LAN.

9.4 DATABASE LIBRARY

The app needs to store messages and other data in a database. In order to simplify implementation, OMiN uses 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.

9.5 ENCRYPTION SCHEME

9.5.1 Requirements

RELATED TO REAL REQUIREMENTS?

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).

9.5.2 Disaster Area Scenario

WHY?

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.

9.5.3 Metro Scenario

WHY?

- In the metro scenario, the following requirements are necessary:
- Verifiable source.
- Verifiable integrity.

- Unbroken encryption scheme.

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

9.5.4 Algorithm Choice

I reviewed 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. Signing a message to verify its origin and integrity is arguably more important than deciding who can read it, so an IBE signing scheme is necessary. We can combine the approaches and use PS06 for signing messages and LW11 for encrypting them. Nodes will store two secret keys – a PS06 key for signing and an LW11 key for decryption. Section 8.5 (page 26) discusses a new design for the cryptography system taking into account that we can only use an IBE scheme for message signing.

9.6 PKG SERVER

The PKG server is a single point of failure, so it has to be very secure. Existing web servers are more secure than building my own, so the OMiN PKG server is written as a CGI script served by a web server. 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.

10 ETHICS

KEEP AND UPDATE IF TESTED WITH ACTUAL USERS

In order to test the real world performance of the network people use.... 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.

11 EVALUATION AND CRITICAL APPRAISAL

11.1 TESTING

TODO: DOES IT CRASH AND BURN?

11.2 OBJECTIVES

TODO: DOES IT MEET OBJECTIVES?

11.3 REQUIREMENTS

TODO: DOES IT MEET REQUIREMENTS?

11.4 USE CASES

TODO: DOES IT MEET USE CASES?

11.5 FURTHER WORK

TODO: STUFF I COULD'T COMPLETE IN TIME

11.5.1 Offline Installation

Currently, installing the app requires downloading it from the app store, which requires an internet connection. This defeats the objective of the network – to communicate without using the internet. There is an alternative method of installation though – the Android installation file (APK) can be sent to another device via Bluetooth, where it can be installed without access to the outside world. This is possible in Android because apps can locate their own APK files using the ApplicationInfoAPI:

<https://developer.android.com/reference/android/content/pm/ApplicationInfo.html#public-SourceDir>.

11.5.2 Reduce Server Overhead

Java is not suited to CGI scripts because of the overhead of starting the JVM for every CGI request. Given that the server is only designed to handle one request at a time (to avoid

security problems due to race conditions), it would be more efficient to have a single JVM instance running constantly with a queue of requests to respond to.

11.6 OVERALL EVALUATION

TODO

12 CONCLUSIONS

TODO

13 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. Pentland A, Fletcher R, Hasson A. DakNet: rethinking connectivity in developing nations. Computer. 2004 Jan;37(1):78-83. [Online]. Available from: <http://dx.doi.org/10.1109/MC.2004.1260729>.
4. 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>.
5. 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>.
6. 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>.
7. Vahdat A, Becker D. ; Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University; 2000.
8. 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>.
9. 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>.
10. 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>.
11. 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>.
12. 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>.
13. 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.](http://dx.doi.org/10.1109/TMC.2003.1195151)
14. 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>.
 15. 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>.
 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.
 24. 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.

CR.YP.TO/BIB/DEVIL-CITE.HTML

URLS FOR EVERYTHING

REMOVE [ONLINE]

14 APPENDICES

14.1 APPENDIX 1 - USER MANUAL

14.1.1 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
```

14.1.2 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.

14.1.3 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*

14.1.4 Installation - 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).

14.1.5 Installation - Android App

TODO: ADD TO APP STORE

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

```
./gradlew installDebug
```

14.1.6 Use

TODO: MAKE UI