OMiN - an Opportunistic Microblogging Network

Neil Wells & Tristan Henderson

ABSTRACT

OMiN is a network designed for scenarios where a direct connection to the outside world (via the Internet or a mobile phone) is impossible, such developing countries, disaster areas and even the subway. It is a pocket switched network – a form of opportunistic network designed to run on smartphones and tablets. OMiN devices store messages and pass them on to other devices via Bluetooth until the message reaches the intended recipient.

This report explores the problem of efficient and secure routing in such a network. This is a hard problem because interactions with other devices are rare and often random. OMiN implements a simple but effective algorithm with lots of potential for expansion.

Messages pass through many (potentially compromised) devices, so the network must be able to guarantee the origin and integrity of a message. Security in opportunistic networks is an unsolved problem because – while most security mechanisms rely on fast access to a central server – opportunistic networks are poorly suited to central servers as messages will take longer to reach the central server as the network scales. This report presents a solution to this problem which only requires one node in the network to have had a connection to the Internet at some point. OMiN partially implements this scheme.

INCLUDE SUMMARY OF EVALUATION

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

The world has become much more connected in recent years due to the ubiquity of the Internet. But there are still huge areas – disaster areas, developing countries and subways – where it is impossible to connect to the outside world. Bringing the internet to these areas requires a lot of infrastructure and money, but a developing technology – the opportunistic network – can provide connectivity to the outside world in an affordable way by using nodes (such as smartphones) which move about to transport messages. This report details OMiN, an opportunistic network using Android smartphones and tablets to transfer messages using Bluetooth. Particular attention has been paid to the problems of routing messages effectively and distributing them securely.

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.

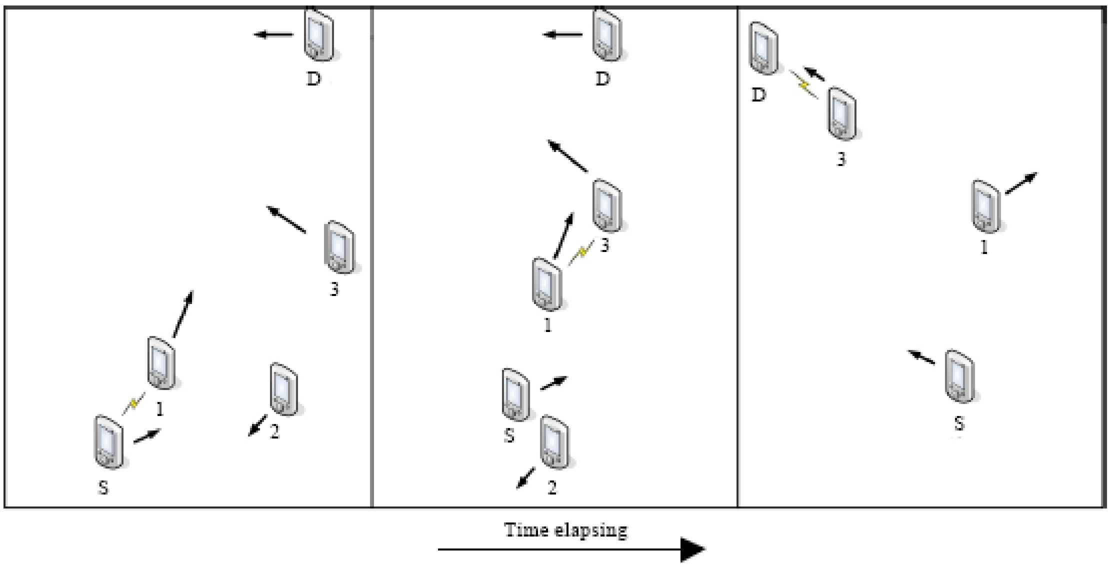
2.1 MICROBLOGGING

A microblogging service allows users to post short messages (microblogs) which can be viewed by others [1]. They are often used as part of a social network – online networks of users, each with a ‘profile’ and set of connections to other users [2]. Twitter’s[1] 140-character tweets are a good example of microblogging in a social network environment. Microblogging services are rapidly gaining popularity because of their ability to spread news quickly and to distil updates into short summaries [1].

2.2 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 [3]. Opportunistic networks have a key advantage over any other form of network – they do not require any infrastructure (cables, towers etc.) to work. While they are very slow compared to other forms of networks, they are often deployed in areas where other forms of network cannot be used because the infrastructure does not exist (such as disaster areas where the infrastructure has been wiped out and developing countries where the infrastructure is too expensive).

A common form of opportunistic network (and the form we will focus on) is the Pocket Switched Network (PSN) – a network of devices (normally smartphones) carried around by people [4]. Connections are made between devices in close proximity using a short range protocol such as Bluetooth.



[1] https://twitter.com

For example, we could send a message from sender S to destination D even when S and D never meet. 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.

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. Because of the predictable nature of human behaviour, much research has been done to improve routing algorithms in pocket switched networks.

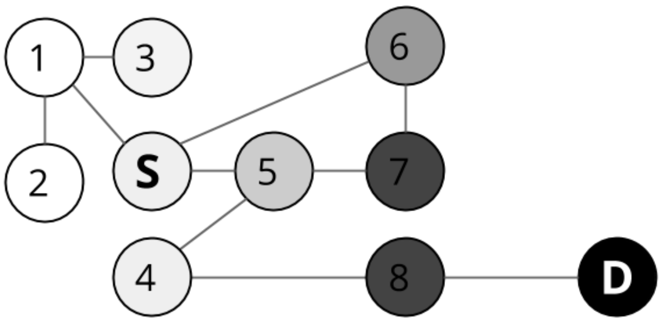
2.3 ROUTING ALGORITHMS

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).

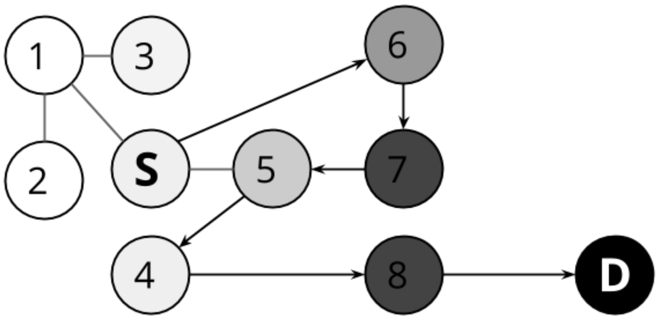
For demonstration purposes, we will use the following network which is trying to pass a message from source S to destination D. The colour of a node represents some heuristic measure of utility (distance to D), where darker nodes are closer to the destination and lighter nodes are further away. I have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



2.3.1 Context-Based Routing

Conventional routing algorithms work by passing a single packet between routers until it reaches its destination. When this approach is taken by an opportunistic network, it is known as 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 – if S was closer to D than node 4 then the message would get stuck in an infinite loop (S-6-5-7-S-6…). 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, context-based routing algorithms use very few resources as the message is never copied, so is a good choice for networks with a good heuristic for node utility (which is why it is used in conventional networks with a predictable structure).

Context-based routing algorithms are very susceptible to blackhole attacks, where a node (known as a blackhole) accepts a message but refuses to pass it on. In this case, the message will never reach its destination. There are heuristics for identifying blackholes such as IRONMAN [7], but none of them can totally protect the network against blackholes, because the only way to identify a blackhole node is for it to drop a message.

If a message has multiple destinations, the sender will have to explicitly create a copy of the message for every destination – this is impossible if there are an unknown number of destinations (e.g. in a publish-subscribe system).

2.3.2 Dissemination-Based Routing

Dissemination-based routing is a paradigm designed to offset the weaknesses of context-based routing in opportunistic networking environments. Instead of passing one copy of the message around, nodes in a dissemination-based network create copies of the message. This is akin to a breadth-first search of the network – if the sender is the root node, copies of the message are passed to the children (nodes who connect directly to the sender) of the sender, who pass it on to their children in turn and so on.

This approach effectively searches multiple possible paths at the same time. If messages are passed on to every node that is encountered, it is guaranteed to find the shortest path if one exists.

While this is a very useful property, it means that every message will be passed to every node on the network and every node must store every message ever sent: this approach is not scalable because nodes in large networks would have to store a large number of messages. For this reason, dissemination-based routing algorithms normally include heuristic for deciding when a message should be passed on.

This approach also creates another problem – when a message reaches its destination, all copies of the message are now pointless. But how does a node know that it can discard its copy? At present, the best solution we have is to discard old messages in favour of new ones.

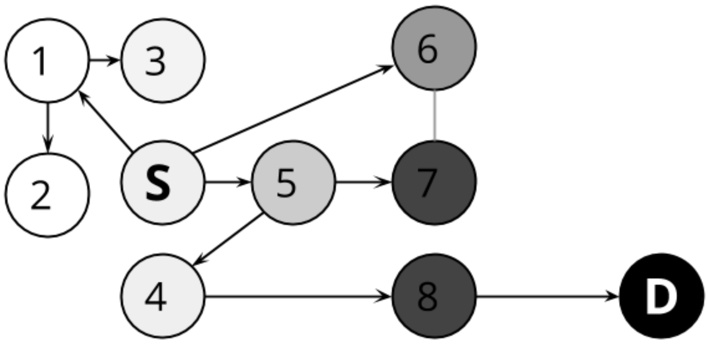
2.3.3 Comparison of Techniques

Given a good heuristic for node utility, context-based routing algorithms are much more efficient than their dissemination-based counterparts: only one copy of a message is stored at any point in time, and the message is removed from the network when it reaches its destination. However they are very susceptible to blackhole attacks and often fail to find a good path to the destination because an opportunistic network structure is very random and cannot be predicted well by heuristics. They cannot be used in networks with an unknown number of destination.

Dissemination-based counterparts are much better at finding a path to the destination and are much less susceptible to blackhole attacks, but require every node to store many more messages than a context-based algorithm would.

2.3.4 Epidemic Routing

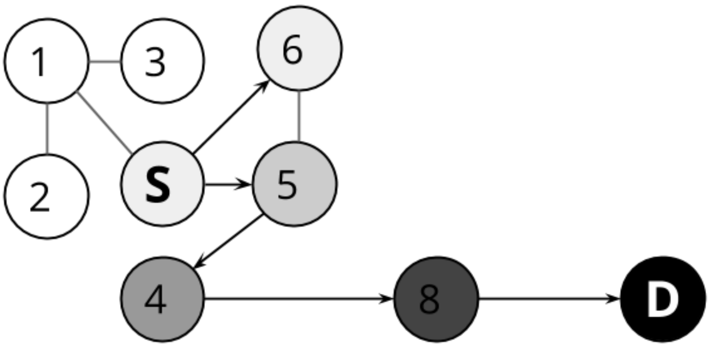
Epidemic routing is the simplest form of dissemination-based routing [8]. Copies of the message are passed at every opportunity until it saturates the network – a method guaranteed to reach the recipient. This is often likened to the spread of a virus.



In the above example, the message reaches the destination by the shortest path, but is also passed to every other node in the network. This approach is undesirable in most cases because of the high resource usage, which is why more advanced dissemination-based algorithms are often used.

2.3.5 PROPHET

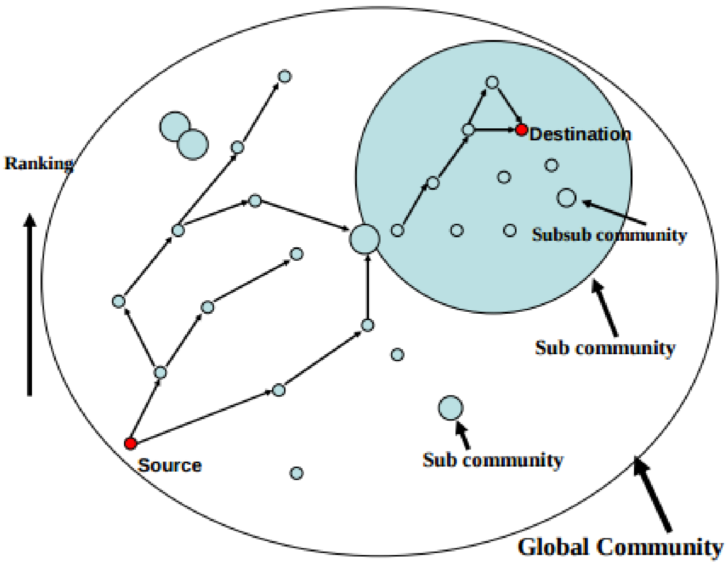
The Probabilistic Routing in Intermittently Connected Networks (PROPHET) algorithm [9] is a common dissemination-based algorithm. The algorithm relies on the heuristic that if a node was encountered recently, it is likely to be encountered again. A node's utility is derived from the utility of recently 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.



PROPHET is considered to be a very good routing algorithm because it will normally find a good path while only sending the message to a subset of the network, making it much more scalable than naïve epidemic routing.

2.3.6 Bubble RAP

The Haggle project discovered that algorithms that treat routing as a generic graph search problem are often unsuited to PSNs. Bubble RAP [10] 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 [10].

2.4 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.

2.4.1 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 [11] or distrusting nodes exhibiting strange behaviour [7].

2.4.2 Cryptographic Security

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

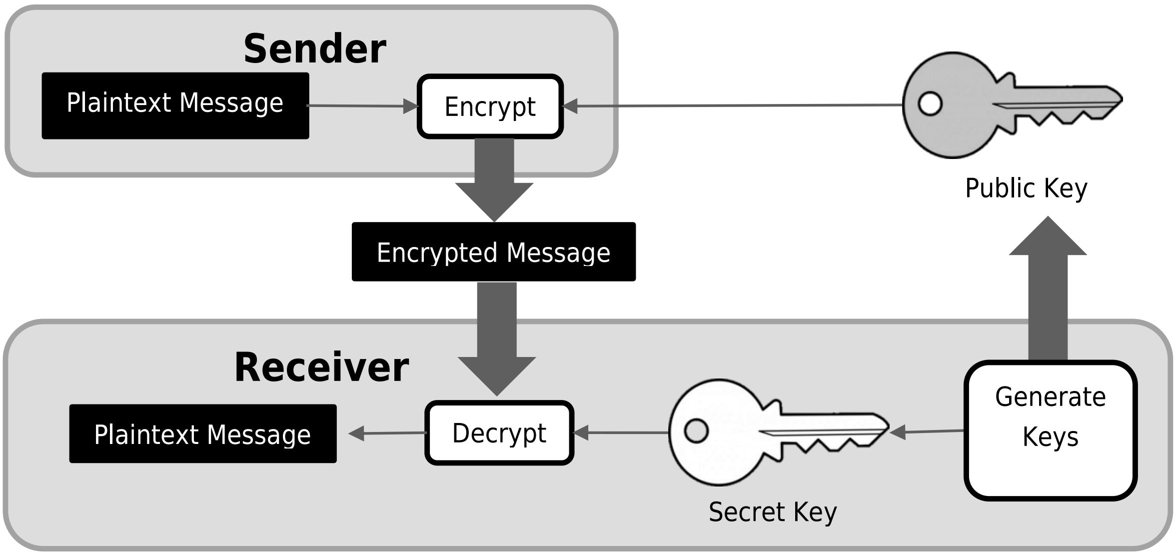
This is infeasible in a scalable opportunistic network because as the network grows, the time to communicate with the central server increases. Some mechanisms, like the one proposed by Shikfa et al [12] do use a central server, but only require it to be available for nodes joining the network. Other 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.

2.4.3 Asymmetric Key Cryptography

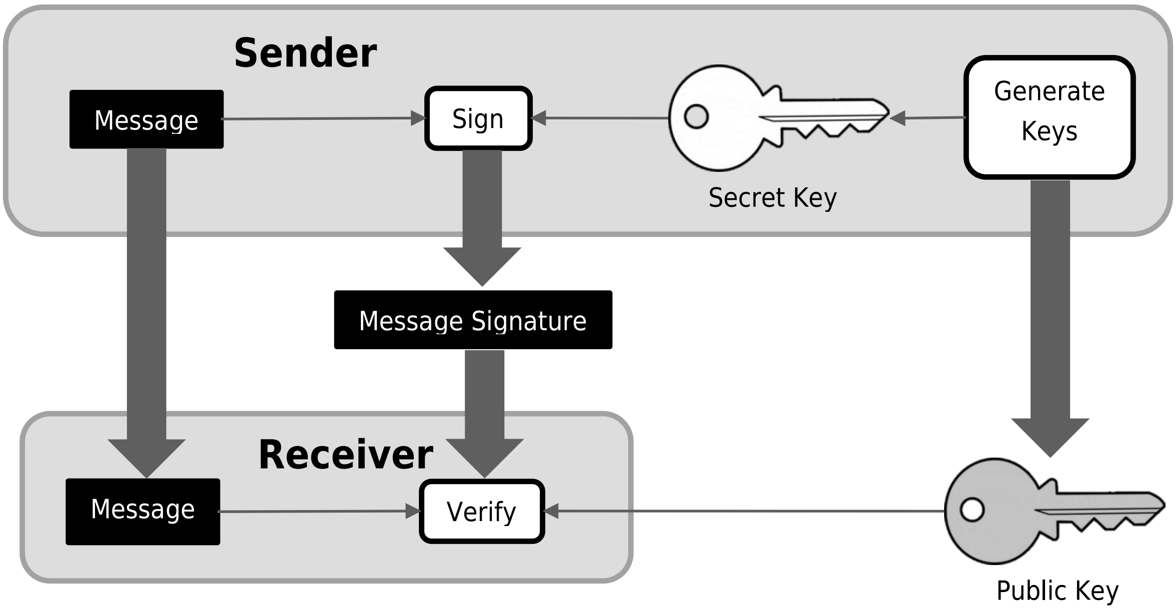
TALK ABOUT CRYPTOGRAPHY INSTEAD OF ENCRYPTION

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.



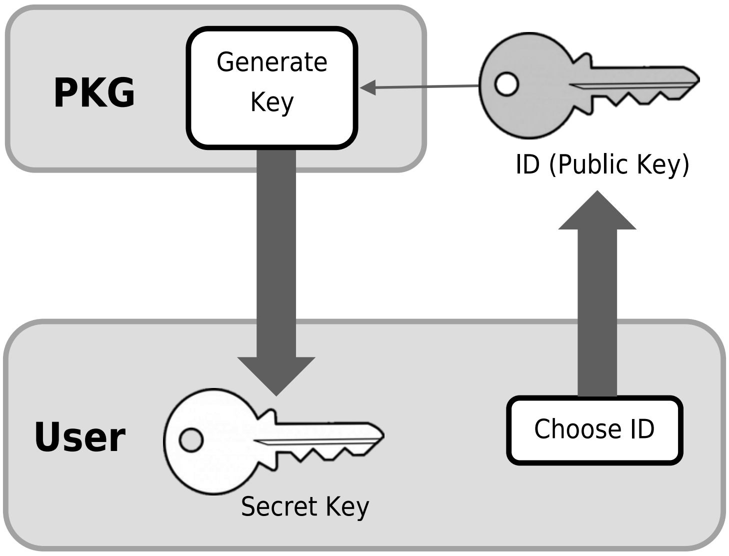
Similarly, the secret key can be used to create a message signature. This signature can be used to verify that the message creator knows the secret key and that the message has not been modified after it was created (data integrity).



2.4.4 Identity-Based Cryptography

REFERENCE

Identity-based cryptography (IBC) 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.



IBC algorithms capable of signing messages are referred to as ID-based signature (IBS) algorithms.

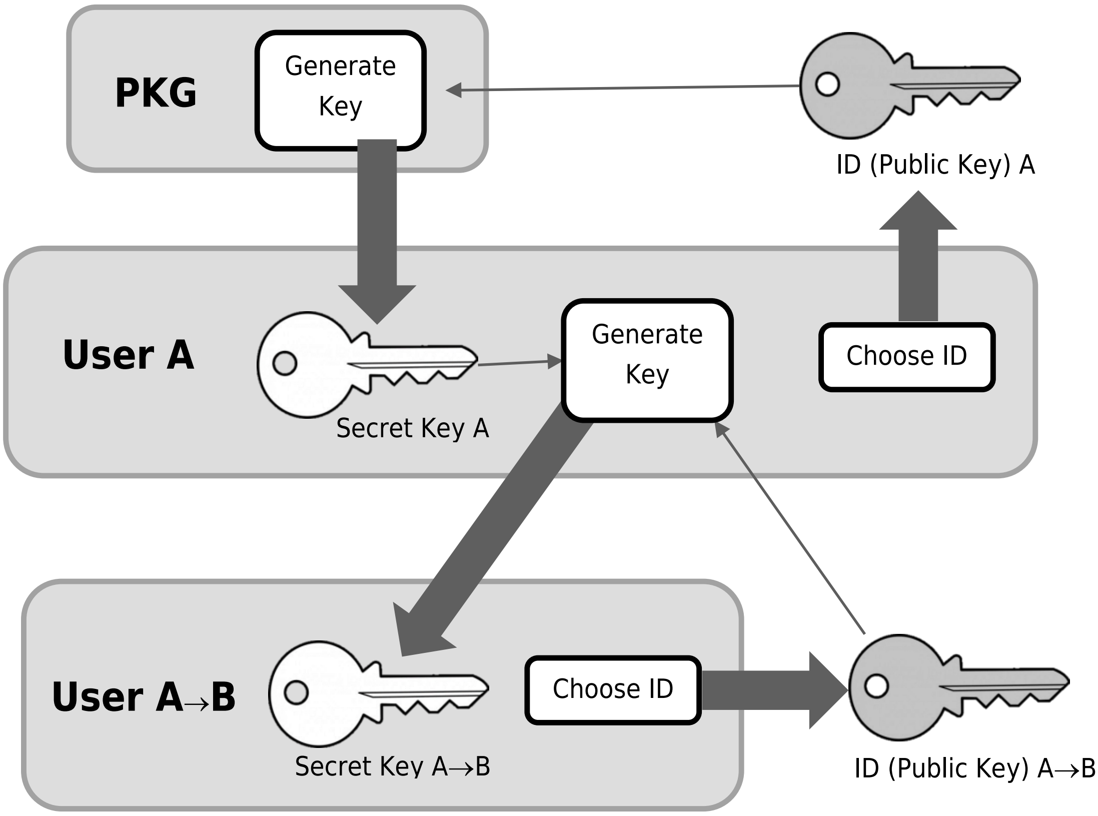
When applied to opportunistic networks, this approach has similar problems to certificate-based approaches - a central server is needed. Some security frameworks assume that there is a central PKG that can and will be accessed occasionally [14]. Others split up the PKG into multiple nodes, some of whom 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).

2.4.5 Hierarchical Identity-Based Cryptography

REFERENCE

TALK ABOUT CRYPTOGRAPHY INSTEAD OF ENCRYPTION

Hierarchical identity-based cryptography (HIBC) is a form of IBC where any node with a secret key can generate a secret key for another node (known as delegation). This creates a tree hierarchy where the central PKG is the root. A use's public key is a combination of their own ID and the public key of the node they received a key from – in other words, a user's public key is the path from the user to the root node (the PKG). For example, the central PKG generates a secret key for a user ID A. User A can now delegate a secret key for users B and C with public keys A→B, A→C respectively.



Again, HIBC schemes which can sign messages are referred to as HIBS (hierarchical identity-based signature schemes.

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

2.4.6 SSNR-OSNR Obfuscation

TODO

[17]

2.5 SIMILAR PROJECTS

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

2.5.1 Haggle

Haggle[2] [18] - 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[3] and Windows Mobile.

[2] http://www.haggleproject.org
[3] http://play.google.com/store/apps/details?id=org.haggle.kernel

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 [4].

2.5.2 FireChat

FireChat[4] is a smartphone application used for off-the-grid messaging between nearby users. It has been used to circumvent government restrictions in Iraq[5] and during the Hong Kong protests[6].

However, the app mostly relies on an Internet connection, and its simple protocol is insecure[7] 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.

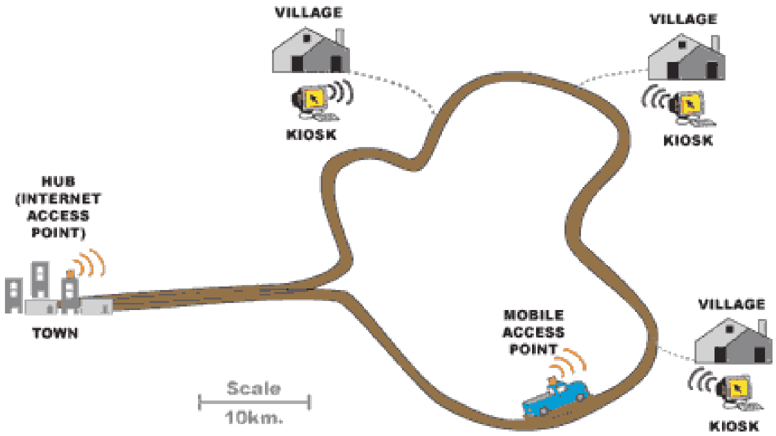
FIRECHAT GREENSTONE - HTTP://WWW.THEVERGE.COM/2015/3/23/8267387/FIRECHAT-GREENSTONE-MESH-NETWORK-BLUETOOTH-WIFI-PEER-TO-PEER

IRAQ/HONK KONG PROTESTS AND INSECURE FIRECHAT ARE REFERENCES

2.5.3 Daknet

Daknet [19] is an opportunistic network in for 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.

[4] http://opengarden.com/firechat
[5] http://www.theguardian.com/technology/2014/jun/24/firechat-updates-as-40000-iraqis-download-mesh-chat-app-to-get-online-in-censored-baghdad
[6] http://www.theguardian.com/world/2014/sep/29/firechat-messaging-app-powering-hong-kong-protests
[7] http://breizh-entropy.org/~nameless/random/posts/firechat_and_nearby_communication



EXPLAIN PIC

2.5.4 SWIM

The Shared Wireless Infostation Model (SWIM) is a proposed opportunistic network to monitor whales [20]. 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.

3 USE CASES

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

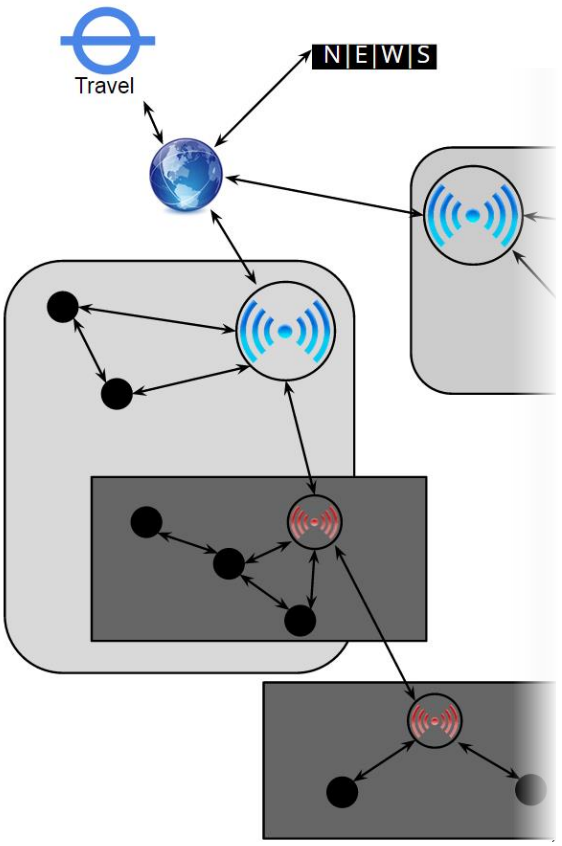
The OMiN network is designed as a general opportunistic network, focusing on the following scenarios where communication over conventional networks is infeasible.

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. 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 cannot be 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: 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 protocol for discovering nodes in close proximity and passing messages and necessary metadata between them.
- Create core library to manage message storage and routing.
- Implement epidemic routing algorithm to send messages to all available nodes.
- Design routing algorithm using user metadata to route messages while disguising message content and metadata.
- Design scheme to allow verification of the origin and integrity of a message.

5.2 SECONDARY OBJECTIVES

- Create user interface.
- Implement more advanced routing algorithm.
- Implement message verification scheme.
- Evaluate impact of message verification scheme.
- Evaluate performance of the implemented routing algorithms.

5.3 TERTIARY OBJECTIVES

- Compare 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 Android device.
- 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.
- Low: The user shall be able to send messages via the Internet to Internet-connected nodes.

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 block 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 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%

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[8], with a backup on my personal GitHub account[9] (the git-remote-hg plugin[10] allows pushing to both Git and Mercurial repositories).

[8] http://ndw.hg.cs.st-andrews.ac.uk/sh-proj
[9] http://github.com/neilw4/OMiN
[10] https://github.com/felipec/git-remote-hg

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.

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 [10] 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 [9] 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.

OMiN messages are microblogs without a specific 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 [21]. 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 [10] [11] 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. 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 does not 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 cryptography 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 HIBS 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 HIBS-BASED APPROACHES

In practice (see the implementation section), there is no HIBS implementation capable of signing messages (although such a scheme is presented in theory by Yuen & Wei [22]. 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.

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. To reduce further instances of this problem later on, nodes should always choose the message signed by the lowest username alphabetically so that 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 [7] and RADON [23] 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.

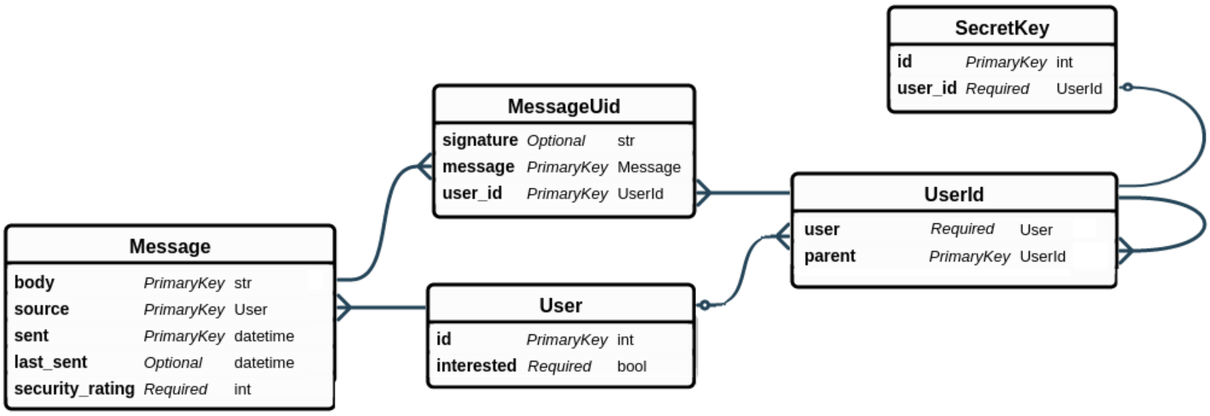
For this reason, the PKG must be built securely. It must be hosted on a secure system, transfer secret keys securely (using SSL) and be invulnerable to injection attacks and unexpected input.

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 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

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 Android devices 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 devices 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 no more efficient than 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 devices being connected or disconnected from a LAN.

LOGENTRIES

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[11] library provided the necessary functionality and was easy to integrate with the application.

9.5 CRYPTOGRAPHY SCHEME

9.5.1 Requirements

RELATED TO REAL REQUIREMENTS?

The cryptography 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 U_1 who authenticates U_2 and so on up to U_x for some arbitrary x.
- Verifiable message source.
- Verifiable message integrity.
- Message contents can be obscured from all but the intended recipient.
- Cryptography scheme will not be broken in the foreseeable future.
- Cryptography 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 cryptography scheme.

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

9.5.4 Cryptography Algorithm Choice

Any cryptography algorithm capable of signing a message can use the signature to verify the message's origin and integrity. Yuen et al. describe the ideal cryptography algorithm [22], a

[11] http://satyan.github.io/sugar

hierarchical ID-based scheme capable of signing messages. Unfortunately I cannot find an implementation of this, or any other HIBS. The best algorithm I can find with an existing implementation is described by Paterson & Schuldt [24], an ID-based signing algorithm. Paterson & Schuldt's algorithm hasn't received much attention so it is hard to determine how secure it is, but it should be sufficient for OMiN. 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 implementation of the cryptography algorithm) must be shared between the server and client. Java is not 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 IMPACT OF CRYPTOGRAPHY

Every benchmark was run 3 times and the average taken. PKG benchmarks were run on the school's lyrane host server and Android benchmarks were run on a Nexus 5 running Android 5.0 and connected to the Eduroam Wi-Fi network.

The first time the PKG runs, it must generate a set of master keys. This takes about 1.9s, but should only happen once, when the network is set up. After this is done, it takes around 2ms to generate a secret key for a user. While the actual key generation does not take long, the Android client sees a delay of about 680ms between choosing a username and receiving a secret key. This is because of the overhead of networking, CGI, starting the JVM for the PKG and PKG security checks. I will explore options to reduce this delay in section 11.6.3.

When a user sends a message, it takes 560ms to sign it to verify its origin. This should have very little effect on the network – it is a relatively small delay compared to the time it will take to deliver a message.

More worryingly, it takes 2.3s to verify the authenticity of a message – an operation that happens every time a message is received. While this is unlikely to affect message latency, it is a significant amount of processing which is rarely necessary. Section 11.6.2 explores ways to reduce this overhead.

Overall, the addition of message signing is unlikely to have any negative effect on message latency. However, it does introduce a significant amount of processing for message verification which will have a negative effect on battery life. Further work should be able to reduce the number of messages that need to be verified in order to preserve battery life as much as possible.

11.2 TESTING

TODO: DOES IT CRASH AND BURN?

11.3 OBJECTIVES

Appendix 2 details which objectives the project met and failed to meet. 8 out of 11 objectives were met and 3 were unmet. In particular, although I designed a routing algorithm based on PROPHET, I did not have time to implement it. This prevented me from assessing its real-world performance.

11.4 REQUIREMENTS

Appendix 3 contains a detailed breakdown of exactly which requirements were met. 22 out of 26 requirements were met and 4 were unmet. 21 out of the 22 high and medium priority requirements were met and 1 (the advanced routing algorithm) was unmet.

11.5 USE CASES

11.5.1 Disaster Area

OMiN should be useable in the disaster use case described in section 3.1 (page 17), provided that it has be pre-installed on user's smartphones.

11.5.2 Metro

OMiN needs a number of improvements before it can work successfully in the subway scenario (section 3.2 – page 17). In particular, it needs a more advanced routing protocol which will prioritise popular messages. OMiN also currently lacks the ability to send and receive messages over the Internet (between news organisations and base stations in subway stations).

11.6 FURTHER WORK

While OMiN works very well, there are a number of enhancements that could be made to make it more suitable for the use cases.

11.6.1 Advanced Routing Algorithm

I didn't have time to implement the advanced routing algorithm I described because the project was more focused on security. OMiN currently uses an epidemic routing algorithm – a more advanced algorithm should greatly increase efficiency and decrease resource usage.

11.6.2 Reducing Verification Overhead

As seen in section 11.1, the policy of verifying every message when it is received will potentially use up the battery because of the significant amount of processing involved. There are a number of policies which could reduce the effects of this.

OMiN already avoids making connections when the battery is low, so it will not receive any messages to verify when the battery is low.

Users will not see every message stored on their device – they will only see messages from users they have registered an interest in. These messages must be verified, but other messages that are only being stored by the device are less important. If these less important messages are not verified, the worst case is that a false message will be passed on to the destination, which will identify that it is false and discard it. This can be counterbalanced if every node verifies a small subset of unimportant messages. Perhaps this should only happen when the device is being charged when power consumption is not an issue.

11.6.3 Faster Server

The Android client sees a delay of about 0.68s between choosing a username and receiving a secret key. Since the actual key generation takes very little time (2ms), the rest is overhead which could be reduced.

This is mostly due to the choice of CGI and Java, which are both inefficient at dealing with requests – CGI because it uses a separate process for every request and Java because the JVM has to boot up for every request. CGI has been made redundant by the more efficient FastCGI protocol, but it currently has no integration with Java. If I had a chance to rewrite the server, I would plug the PKG server into a lightweight Java HTTP server such as Jetty. This would allow requests to run in separate threads. These could use Java's in-memory locking structures instead of Unix file locking.

11.6.4 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[12].

11.6.5 Multimedia Messages

There is no reason why OMiN should be restricted to short text messages. Its capabilities could be expanded to allow pictures and formatted text – this would be very useful for distributing news in the Metro use case. The routing algorithm and buffer eviction strategy may have to be modified to consider the size of a message. From a security perspective, we would have to make sure that multimedia messages cannot be used by an attacker to gain control over a device.

11.6.6 Internet-Supplemented Networks

OMiN is designed to work in areas with no access to the Internet. However, in cases where the Internet could be used, it could be used to send messages to other internet-connected nodes, reducing message latency. This would be especially useful in the Metro scenario where internet-connected base stations could be used to speed up message distribution and allow messages to be sent from other online nodes.

11.7 OVERALL EVALUATION

TODO

12 https://developer.android.com/reference/android/content/pm/ApplicationInfo.html#publicSourceDir

12 CONCLUSIONS

TODO

13 BIBLIOGRAPHY

1. Kaplan AM, Haenlein M. The early bird catches the news: Nine things you should know about micro-blogging. Business Horizons. 2011;54(2):105-113. Available from: http://dx.doi.org/10.1016/j.bushor.2010.09.004.

2. Boyd D, Ellison NB. Social Network Sites: Definition, History, and Scholarship. Journal of Computer-Mediated Communication. 2007;13(1):210-230. Available from: http://dx.doi.org/10.1111/j.1083-6101.2007.00393.x.

3. Kaur N. AN INTRODUCTION TO WIRELESS MOBILE SOCIAL NETWORKING IN OPPORTUNISTIC COMMUNICATION. International Journal of Distributed & Parallel Systems. 2013;4(3). Available from: http://airccse.org/journal/ijdps/papers/4313ijdps06.pdf.

4. 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. Available from: http://dx.doi.org/10.1109/TMC.2007.1060.

5. Musolesi M, Hailes S, Mascolo C. ; 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. ; 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. Bigwood G, Henderson T. ; 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.

8. Vahdat A, Becker D. ; Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University; 2000. Available from: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.34.6151.

9. Lindgren A, Doria A, Schelén O. ; 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.

10. Hui P, Crowcroft J, Yoneki E. ; 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.

11. Trifunovic S, Legendre F, Anastasiades C. ; 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.

12. Shikfa A, Onen M, Molva R. ; 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. ; 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.

14. Kamat P, Baliga A, Trappe W. ; 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. ; 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. ; 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. ; Privacy-enhanced social-network routing. Computer Communications. 2012;35(1):62-74. Available from: http://www.sciencedirect.com/science/article/pii/S0140366410004767.

18. 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. Available from: http://hal.inria.fr/inria-00001012.

19. Pentland A, Fletcher R, Hasson A. DakNet: rethinking connectivity in developing nations. Computer. 2004 Jan;37(1):78-83. Available from: http://dx.doi.org/10.1109/MC.2004.1260729.

20. Small T, Haas ZJ. ; 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.

21. BH, Bloom. ; 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.

22. Yuen TH, Wei VK. Constant-Size Hierarchical Identity-Based Signature/Signcryption without Random Oracles. IACR Cryptology ePrint Archive. 2005;2005:412. Available from: http://eprint.iacr.org/2005/412.pdf.

23. Li N, Das SK. ; 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.

24. Paterson KG, Schuldt JCN. ; 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.

14 APPENDICES

Appendix 1 USER MANUAL

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
```

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.

Building

Executing the following command from the project directory will build everything, downloading libraries, build scripts and the Android SDK if necessary:

```
./gradlew build
```

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

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).

Installation - Android App

TODO: ADD TO APP STORE

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

```
./gradlew installDebug
```

Use

TODO: MAKE UI

Appendix 2 OBJECTIVES MET

Importance	Description	Met
Primary	Design and implement protocol for discovering nodes in close proximity and passing messages and necessary metadata between them.	Met
Primary	Create core library to manage message storage and routing.	Met
Primary	Implement epidemic routing algorithm to send messages to all available nodes.	Met
Primary	Design routing algorithm using user metadata to route messages while disguising message content and metadata.	Met
Primary	Design scheme to allow verification of the origin and integrity of a message.	Met
Secondary	Create user interface.	Met
Secondary	Implement more advanced routing algorithm.	Unmet
Secondary	Implement message verification scheme.	Met
Secondary	Evaluate impact of message verification scheme.	Met
Secondary	Evaluate performance of the implemented routing algorithms.	Unmet
Tertiary	Compare real world vs. simulated performance of the routing algorithms.	Unmet

Appendix 3 REQUIREMENTS MET

Met Requirements

Type	Priority	Description
Functional User	High	The user shall be able to use the network on their Android device.
Functional User	High	The user shall be able to create a unique identity.
Functional User	High	The user shall be able to send plain text messages to all others who follow the user.

Functional User	High	The user shall be able to 'follow' any user and receive messages sent by that user.
Functional User	High	The user shall be able to send messages without requiring an Internet connection.
Non-Functional User	Medium	The user shall be able to be confident in the origin and integrity of a message.
Non-Functional User	Low	The user shall be able to use the network with minimal training.
Functional System	High	The system shall work on portable electronic devices such as smartphones or tablets.
Functional System	High	The system shall allow creation of user identities with a unique cryptographic identity.
Functional System	High	The system shall automatically connect to nearby nodes and pass on relevant information.
Functional System	High	The system shall pass on messages until they reach their destination.
Functional System	Medium	The system shall ensure that messages cannot be modified in transit or that such modifications can be detected.
Functional System	Medium	The system shall ensure that nodes cannot send a message that appears to be from another user.
Functional System	Medium	The system shall restrict the size of the message store.
Functional System	Medium	The system shall protect against Sybil attacks.
Functional System	Medium	The system shall prevent messages from being modified while in transit.
Functional System	Medium	The system shall protect user metadata from all other nodes.
Non-Functional System	High	The system shall work in an unstructured environment with random encounters between nodes.
Non-Functional System	High	The system shall not require a connection to any other network (such as the Internet).
Non-Functional System	Medium	The system shall be robust and able to continue functioning when it encounters an unexpected state such as a malfunctioning or untrustworthy node.
Non-Functional System	Medium	The system shall minimise the number of messages lost before they reach their destination.

Non-Functional System	Medium	The system shall deliver messages as quickly as possible.

Unmet Requirements

Type	**Priority**	**Description**	**Comments**
Functional User	Low	The user shall be able to send messages via the Internet to Internet-connected nodes.	Not implemented – would require a lot of work.
Functional System	Low	The system shall have mechanisms to mitigate Denial of Service attacks.	Not implemented.
Functional System	Low	The system shall block attempts to prevent message propagation.	Blackhole prevention was not implemented but the dissemination-based routing protocol makes it hard for blackholes to block message.
Non-Functional System	Medium	The system shall route messages effectively given a semi-predictable set of encounters between nodes.	Non-epidemic routing protocol designed but not implemented.