Privacy-enhanced social network routing in opportunistic networks

Iain Parris, Greg Bigwood and Tristan Henderson
School of Computer Science
University of St Andrews
St Andrews, Fife, KY16 9SX, UK
{ip,gjb,tristan}@cs.st-andrews.ac.uk

***Abstract*—Opportunistic networking — forwarding messages in a disconnected mobile ad hoc network via any encountered nodes — offers a new mechanism for exploiting the mobile devices that many users already carry. Forwarding messages in such a network often involves the use of social network routing— sending messages via nodes in the sender or recipient's social network. Simple social network routing, however, may broadcast these social networks, which introduces privacy concerns.**

This paper introduces two methods for enhancing privacy in social network routing by obfuscating the social network graphs used to inform routing decisions. We evaluate these methods using two real-world datasets, and find that it is possible to obfuscate the social network information without leading to a significant decrease in routing performance.

I. Introduction

Mobile devices, such as mobile phones, are commonly carried by people. While most current communication using such devices takes place through infrastructure such as licensed GSM or UMTS networks, it may be possible to exploit these devices in an ad hoc manner. By directly exchanging messages between devices when in physical proximity, an *opportunistic network* may thus be formed; messages are sent via intermediary devices, in a disconnected store-and-forward architecture.

One main challenge in opportunistic networks is routing: given episodic connectivity based on people's real-world movements, how can we send messages from source to destination? One approach is *epidemic routing* — flooding the network with messages, by sending messages during each and every encounter [14]. This approach ensures that, if a path exists between source and destination, the message will be delivered along this path as quickly as possible. But sending large numbers of redundant messages is wasteful, and will rapidly drain the mobile devices' batteries.

To reduce message delivery cost, messages should be selectively forwarded during encounters between members of the opportunistic network. What is a good method of determining whether a message should be forwarded?

One approach is *social network routing*. Based on the assumption that encounters between mobile devices are more likely to occur between people in the same social network — i.e., between people who are connected to each other, perhaps through friendship or co-location, than between random strangers — messages may be forwarded selectively only within the sender's social network.

But one problem with social network routing is that of privacy. In social network routing schemes, intermediate nodes forward messages based on whether the encountered node is in the original sender's social network. Social network routing may involve broadcasting social network information in the clear (not encrypted end-to-end because the information is used by intermediate nodes for routing), creating potential privacy concerns. For example, opportunistic network users might wish to hide an embarrassing friend. Or a user may accept the use of social network information for routing, but not the whole network being world-viewable; it is one thing for a curious person to be able to infer some of the social network based on forwarded messages, but another to broadcast the potentially-sensitive information.

Our goal is to mitigate privacy concerns while retaining the advantages of social network routing. In this paper, we:

- Analyse the potential privacy threats implicit in social network routing, to present an attack tree.
- Investigate the effect on routing performance of obfuscating social network graphs.
- Investigate hiding social network information using one-way hashing, via the Bloom filter data structure.

Our contributions are to provide what is, to our knowledge, the first analysis of threats in social network routing, and the first schemes to attempt to enhance privacy in social network routing without key management.

We discuss related work in the next section, and present a threat analysis in Section III. We discuss our two social network routing schemes in Section IV. Section V evaluates these schemes using two real-world traces, and finally in Section VI we conclude and discuss ongoing work.

II. Related work

Opportunistic networks [11] have become increasingly popular and relevant as more people carry mobile devices. Essentially, an opportunistic network is a disconnected MANET (mobile ad hoc network), where mobile nodes can send messages in the absence of any knowledge about

978-1-4244-5328-3/10/$26.00 ©2010 IEEE

network topology. Nodes opportunistically make use of any other nodes that they encounter, as long as these encountered nodes are likely to help the message reach its destination.

The performance of an opportunistic network depends on accurately determining which encountered nodes will be useful in forwarding. Many forwarding schemes have been proposed that leverage the structure of nodes' social networks to do so [5], [9]; if we know that a node is in the same social network as a message's destination, then it may make sense to use that node for forwarding. In this paper, we refer to this class of opportunistic network forwarding protocols as *social network routing,* and the class of protocols which broadcast social network information in the clear as *simple social network routing.*

If nodes are to trust their data with any other nodes that they encounter, privacy is paramount. Existing proposals for addressing privacy in opportunistic networks, e.g., [4], [13], use key management to divide network users into groups and restrict access accordingly. Key distribution and management in such schemes is very difficult in an ad hoc environment, however, and may impede the very feature which makes opportunistic networking so appealing — the fact that nodes may forward to *any* node that they encounter. Moreover, even within these systems, group members can observe the routing tables of all other members, so many of the attacks that we describe in this paper are still possible.

Aad et al. [1] present methods to improve anonymity within an ad hoc network. These include using Bloom filters to compress and obscure a packet's routing list, and a technique for combining multicast and onion routing. They, however, assume global routing information is available for the network, which we do not; and they do not evaluate performance using simulations, as we do here.

III. Threat analysis

Before we can enhance privacy in social network routing, we need to understand the threats against privacy that may occur when using such routing schemes. We choose to employ *attack trees*, as introduced by Schneier [12].

An attack tree is a type of *and–or tree*, used to enumerate attacks against a system. The root node of the tree is the overall attack goal, while nodes within the tree are subgoals. The children of a particular node are the steps required to achieve that node's subgoal. By constructing such a tree from the root node (overall goal) downwards, we may enumerate a structured threat analysis for attacks against a system.

Following is a preliminary attack tree for privacy threats against users of opportunistic networks employing social network routing. Our attack tree may not be complete; future work is to further increase the rigour of our threat analysis.

A. Goal: Discover structural information about the social network graph.

1) Learn whether a friendship link exists (or does not exist) between two users. OR
    a) Discover communication (or lack of) between the users. OR
        i) Eavesdrop a message as it is forwarded user-to-user, from source to final destination (or any intermediary). OR
            A) In simple social network routing, a message traced along such a path reveals social network links (or lack of) – because messages are forwarded if and only friendship links exist. Friendship links are the path traversed by the message.
        ii) Extract source/destination from an intercepted message to an intermediary.
    b) Extract friendship links from an intercepted message to an intermediary.
2) Learn how many friendship links a particular user has.
    a) Extract friendship links from an intercepted message to an intermediary.

B. Goal: Discover whether two individuals have been in proximity within a certain timeframe.

1) Follow one or both individuals for the time in question. OR
2) Infer proximity by sending a specially crafted message, and making inferences based on where the message is observed within the network. OR
    a) Example: has Alice from New York recently met Bob from Los Angeles? To find out, an attacker Mallory in New York can inject a message addressed for colluding attacker Trudy in Los Angeles into the system, with Alice and Bob only as requested intermediaries. If Trudy receives Mallory's message, Mallory and Trudy have learned that Alice and Bob have met within the lifetime of this malicious message.
3) Infer proximity by noting that messages are not forwarded twice. OR
    a) Example: if a message is not forwarded to a node known to be a requested intermediary, the message must already have been forwarded earlier. An attacker can infer that the nodes were in proximity before this time. This is a passive version of 2.
4) Wait in a common place and listen for message traffic. Message exchange, or message headers, may reveal the colocation of individuals to an attacker.

C. Goal: De-anonymise a social network to discover the presence of individuals within the network.

1) Follow individuals, and tie their network identifiers to their actual identities. OR
2) Infer identities from known portions of the social network.

a) Example: if five people are known to be mutual friends, and four are deanonymised with a fifth mysterious node, an attacker can infer that this unknown node is the last member of the clique.

IV. Privacy-enhanced social network routing

In simple social network routing schemes, the sender's social network is transmitted in the clear along with each message. Intermediate forwarding nodes are able to read the sender's full social network in plaintext, facilitating most of the threats outlined in Section III.

Encrypting the social network information end-to-end can ensure privacy, but we would then lose the advantages of social network routing: intermediate forwarding nodes would no longer be able to exploit the sender's social network information to inform their routing decisions.

Inspired by [2], we attempt to target the social network routing privacy threats by obfuscating a sender's social network. We now introduce two schemes for doing so.

A. Statisticulated Social Network Routing

Named for a portmanteau of *statistical manipulation*[1], our first scheme is *Statisticulated Social Network Routing* (*SSNR*). For each message transmitted, the sender makes changes to the message's copy of their social network — adding or removing nodes. While the social network sent along with the message will be based to some extent on the sender's true social network, and so still useful for social network routing, the social network has been modified by the addition or removal of nodes. Any node seeing the social network sent along with the message now cannot say with certainty whether a particular node is truly part of, or absent from, the sender's social network.

In practice, the sender may choose the level of social network manipulation on a per-message basis. In our evaluation, however, we examine routing performance for a particular choice of modification degree of the sender's social network. For instance, a +50% modification of the social network would meant that the sender adds 50% more nodes to their social network before message transmission. We thus determine average performance for a particular degree of social network modification. For simplicity, we do not evaluate routing performance while simultaneously adding and removing nodes.

It would still be possible for a malicious person to average over the social network information included with many messages of one particular sender. But we have created much more work for this malicious person: many generated messages must be intercepted, rather than just one single message to reveal all. Since the nodes in the opportunistic network are mobile, and messages are only transmitted when nodes encounter one another, a malicious person would likely have to physically follow a node for some length of time before being able to intercept multiple messages from the same source; a task considerably more challenging than eavesdropping a single message.

[1]Huff coins the term *statisticulation* in [8]: "Misinforming people by the use of statistical material might be called statistical manipulation; in a word (though not a very good one), statisticulation."

B. Obfuscated Social Network Routing

Our second scheme, *Obfuscated Social Network Routing* (*OSNR*), embeds the social network information within a Bloom filter. A Bloom filter [3] is a data structure allowing probabilistic querying for set membership. False negatives are not possible, but false positives are — with increasing probability as the Bloom filter becomes more full. After inserting each node in the sender's social network into a Bloom filter, we may regard the Bloom filter as a non-trivially-reversible hash of this social network information.

To make a rainbow table attack[2] impractical, we create a per-message random salt, which is sent along with the message in the clear. The elements inserted into the Bloom filter are a concatenation of this random salt with a unique node identifier (any unique identifier would suffice, such as MAC address, IMEI, or even some higher-level identifier tied to the user rather than the device).

Given the Bloom filter, the random salt and an encountered node's identifier, it is easy to make a routing decision: query for set membership of the random salt concatenated with the node identifier. A positive result — guaranteed if the node is inside the sender's social network, but possible with low probability if not — means to forward the message, since the node is most likely in the sender's social network. A negative result means that the node is not in the sender's social network, and so not to forward.

Since we do not employ encryption (given the lack of a PKI), it is still perhaps possible for an attacker to reverse engineer the Bloom filter by brute force — the attacker can iterate through all the node identifiers, concatenating each with the plaintext salt and testing for a Bloom filter match. This is orders of magnitude more work than a rainbow table lookup, however, and must be repeated for every message. The Bloom filter (with salt) does not provide perfect security, but does make the attacker's job much harder.

It is possible to combine *OSNR* and *SSNR*: the social network may be modified as in *SSNR* prior to hashing in a Bloom filter as in *OSNR*. We refer to this as *SSNR-OSNR*.

We note that Bloom filters are fixed-width — a convenient property for scalability. In pure *SSNR*, packet headers may grow arbitrarily large as the sender's social network grows; this is potentially a problem for sender with very large social networks (and compounded if these networks grow further using *SSNR*). *OSNR*, and *SSNR-OSNR*, have no such scaling problem due to the fixed size of the Bloom filter.

[2]A rainbow table is a precomputed lookup table of hash value to input.

V. Evaluation and Results

We now evaluate our two schemes to determine their impact on opportunistic network performance. We use trace-driven simulation with two real-world datasets.

A. Datasets

We collected the first dataset — which we call the *SASSY* dataset — in a previous experiment. 25 participants were equipped with 802.15.4 Tmote Invent sensor motes and encounters were tracked for a period of 79 days, from which we selected a 30-day section for our simulations.

The original dataset was very sparse due to hardware limitations which meant that many encounters were lost. Inspired by [7], we augment our traces using a working-day and augmented random-waypoint model. Nodes randomly select a waypoint from a set of points of interest and walk according to predetermined paths (such as roads) to reach these points. Nodes moved at 0.5–1.5ms^{-1}. At each waypoint the nodes could stop for 0–120s. Each node was additionally randomly assigned a home location, and the nodes would travel to this location to "sleep" for 8 hours in every 24. Each node had an additional 10% chance of either choosing to go to the Computer Science departmental buildings (since our participants were mainly Computer Science students) or their "home" at any waypoint selection.

The social network information for the *SASSY* dataset was self-reported by the 25 participants at the start of the experiment: their Facebook "friends". Many participants knew each other — the mean number of other participants in each participant's social network (i.e., Facebook friends) was 9.8, with a standard deviation of 5.0.

The second dataset used was the well-known *Reality Mining* (*RM*) dataset collected at MIT [6]. This dataset comprises Bluetooth encounter traces from ≈100 mobile phone users over the course of an academic year. To obtain social network information for this dataset, we use the participants' address book information — if a pair of nodes encounter one another, and at least one has the other in their address book, then each node is said to have the other in its social network. Unlike the *SASSY* dataset, few participants knew each other: 52 participants had at least two participants in their social network (and were thus candidate nodes for our simulations). Of these 52 participants, the mean size of the social network was 3.7, with a standard deviation of 2.0.

As participants left the experiment throughout the year (and new participants joined), we could not treat the dataset as one contiguous trace. We thus select out 30-day segments.

B. Simulation parameters

We performed trace-driven simulations using these two datasets with the following parameters: simulation length of 30 days; 30 messages generated per day; message TTL of one day; at least 10 runs for each set of parameters; *SSNR* obfuscation from −80% to +200% at 20% intervals.[3]

For the *SASSY* dataset, which contains location information, we used a customised version of the ONE simulator [10], which included our augmented random waypoint model, to generate ns-2 traces. For speed, we used ns-2 rather than ONE for all of the simulations. The *RM* dataset has no location information so we could not use ns-2; we instead parsed the Bluetooth encounters and simulated message-passing with a Python program.

C. Performance metrics

We evaluate our simulations using three metrics [9]:

- *Delivery ratio:* proportion of delivered messages, out of the total number of unique messages created.
- *Delivery cost:* total number of messages (including duplicates) transmitted, normalised by the total number of unique messages created.
- *Delivery delay:* time taken for a message to reach its destination.

D. OSNR implementation

Our *OSNR* implementation used a 128-bit Bloom filter. To insert each element (node ID concatenated with a random salt, as described in IV-B) into the filter, the element's 128-bit MD5 hash[4] was divided into four 32-bit integers. Taking each integer mod 128 (the filter length) resulted in four values in range 0-127, and the four corresponding bits in the Bloom filter were, if not already 1, set to 1.

E. Results

Figures 1–6 show our trace-driven simulation results for our routing schemes with the *SASSY* and *RM* datasets. For every set of parameters for our three metrics, *applying* OSNR *did not significantly impact routing performance* — the error bars overlap for each datapoint. Any impact from Bloom filter false positives is so slight as to be insignificant.

Figure 1 shows that for the *SASSY* dataset, delivery ratios are high for all tested social network size target modifications. It is possible to remove 60% of the sender's social network's nodes while still retaining a delivery ratio of over 90% of the ratio with an unmodified social network.

Although much noisier, and with lower delivery ratios, Figure 5 shows a similar result for the *RM* dataset. Large modifications to the size of the sender's social network can be made without significantly affecting the delivery ratio.

[3] If we reach the upper bound of all nodes added, or the lower bound of only one node remaining in the sender's social network, we stop adding or removing nodes for this message.

[4] MD5 is not collision-resistant, but we use the uniformity and one-way properties, not collision-resistance property, of MD5. A maliciously-generated collision does not affect the security of our system, since senders generate Bloom filters on a per-message basis and the ability to generate a collision would merely mean another false positive in routing — which already may occur, and which can much more easily be produced by the malicious sender setting more Bloom filter bits to 1.

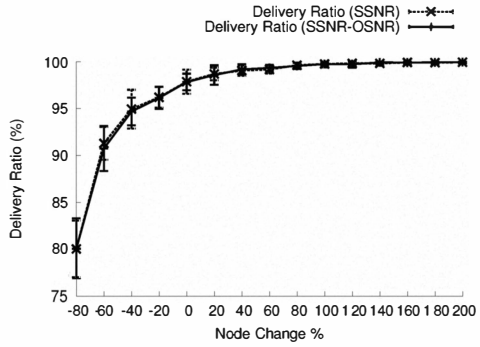

Figure 1. *SASSY* dataset. Delivery ratio vs target percentage modification of each message sender's social network. Error bars indicate 95% confidence intervals. It is possible to remove over half of the social network links while still retaining high message delivery ratios. 98% of messages arrive with simple social network routing. 91% of messages arrive even after removing 60% of the source node's social network links.

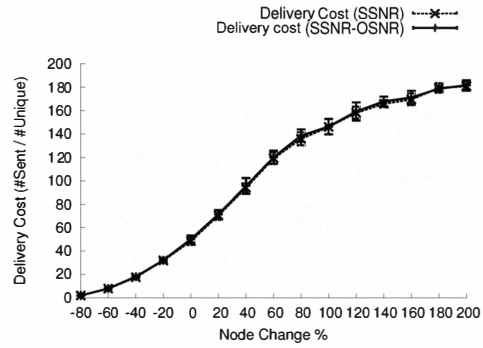

Figure 2. *SASSY* dataset. Message delivery cost vs target percentage modification of the number of friends of each message's original sender. As we obfuscate the sender's social network by adding links, the delivery cost increases.

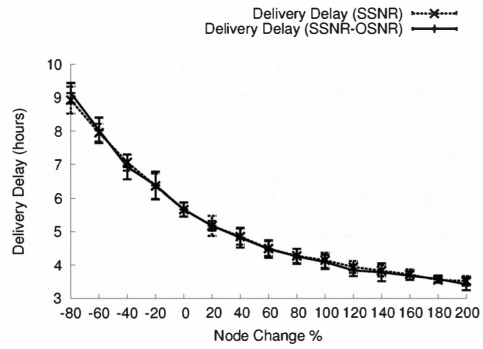

Figure 3. *SASSY* dataset. Message delivery delay vs target percentage modification of each message sender's social network. As we remove from the sender's social network, delivery delay increases – but only from about 6 to 8 hours for simple social network routing compared to *SSNR* with -60% sender social network size target change.

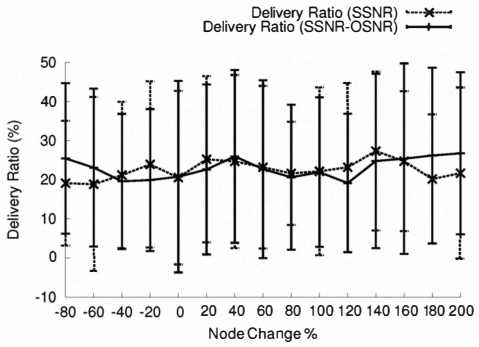

Figure 4. *RM* dataset. Delivery ratio vs target percentage modification of each message sender's social network. It is possible to modify the sender's target social network size greatly (-80%, +200%) without significantly affecting delivery ratio.

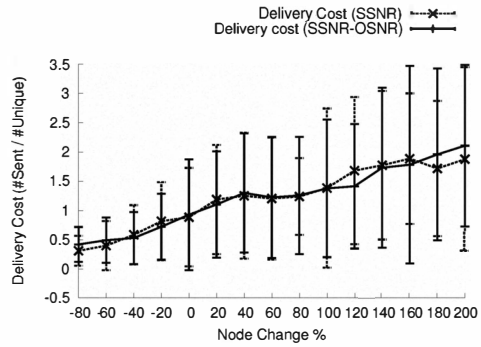

Figure 5. *RM* dataset. Message delivery cost vs target percentage modification of each message sender's social network. As we obfuscate the sender's social network by adding links, the delivery cost increases.

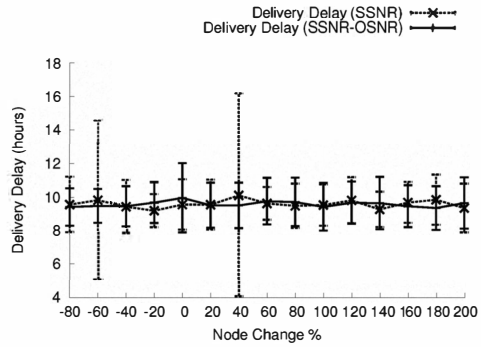

Figure 6. *RM* dataset. Message delivery delay vs target percentage modification of each message sender's social network. The impact on delivery delay when modifying the sender's target social network size is insignificant.

Figure 2 shows delivery cost for the *SASSY* dataset is significantly affected by modifying the sender's social network size: the smaller the social network, the lower the cost of sending a message. Compared to simple social network routing, with 50 data messages per unique message, a -60% change in sender social network results in only 10 data messages: five times fewer. *SSNR* has improved delivery cost, yet simultaneously retained a good delivery ratio (Figure 1) and increased the sender's privacy by not revealing some of their true friends.

Figure 5 shows that delivery cost for the *RM* dataset appears to show a similar trend as for the *SASSY* dataset, but again with more noise. The corresponding absolute figures for delivery cost, however, are lower than *SASSY*— perhaps because *RM* encounters are much sparser.

Figure 3 shows that delivery delay for the *SASSY* dataset increases when removing nodes from the sender's social network. This increase is from $\approx$6 to $\approx$8 hours from simple social network routing to *SSNR* with removing 60% of the sender's social network. If delivery delay is a concern, we may indeed reduce the delay by adding nodes with *SSNR*.

Figure 6, however, shows little correlation between delivery delay and the modification of the size of the sender's social network for the *RM* dataset: any difference that may exist seems to be lost in the noise from this dataset.

Finally, we see that for both datasets we can significantly modify the sender's social network size (e.g., by -60%), thus increasing the privacy of the sender, and yet retain good routing performance. Removing nodes may significantly reduce delivery cost — a beneficial side effect — while enhancing privacy. Conversely, if delivery delay or ratio is paramount, *SSNR* allows adding nodes to improve performance by these metrics, again while enhancing privacy, though at the expense of increased delivery cost.

VI. Conclusions and Future Work

In this paper we have presented two schemes for enhancing privacy in social network routing in opportunistic networks. We find that it is possible to obfuscate a sender's social network by removing up to 60% of the nodes from the social network, while still maintaining a delivery ratio of 90% of unaltered social network routing. We demonstrated that, by using Bloom filters, we can prevent eavesdropping of social network information with only a minimal effect on network performance. We evaluated these two schemes using two real-world datasets. Although the datasets vary widely (including in scale, location and connectivity), our findings appear to hold for both.

We have presented only an initial evaluation of our routing schemes; our work is ongoing. We need to formally analyse whether the *SSNR* and *OSNR* schemes provide consistent deniability. We are exploring refined versions of these schemes, e.g., selecting popular or well-connected nodes to remove or add to a node's social network, although these may introduce additional attacks. We are also exploring more of the attacks described in our threat analysis, and testing against our schemes in simulation.

We note that the two datasets used to evaluate our routing schemes may not be representative of general opportunistic network usage. Both involve participants who opted in to small-scale experiments; their social networks and mobility patterns may differ from a universal deployment. We are searching for new datasets to use to evaluate our schemes.

References

[1] I. Aad, C. Castelluccia, and J.-P. Huubaux. Packet coding for strong anonymity in ad hoc networks. In *Proc. Securecomm*, pp 1–10, Baltimore, MD, USA, 2006.

[2] S. K. Belle and M. Waldvogel. Consistent deniable lying: Privacy in mobile social networks. In *Proc. Workshop on Security and Privacy Issues in Mobile Phone Use*, Sydney, Australia, 2008.

[3] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Comm. of the ACM*, 13(7):422–426, 1970.

[4] C. Boldrini, M. Conti, and A. Passarella. Exploiting users' social relations to forward data in opportunistic networks: The HiBOp solution. *Pervasive and Mobile Computing*, 4(5):633–657, 2008.

[5] E. M. Daly and M. Haahr. Social network analysis for information flow in disconnected delay-tolerant MANETs. *IEEE Trans. on Mob. Comp.*, 8(5):606–621, 2009.

[6] N. Eagle, A. S. Pentland, and D. Lazer. Inferring friendship network structure by using mobile phone data. *PNAS*, 106(36):15274–15278, 2009.

[7] F. Ekman, A. Keränen, J. Karvo, and J. Ott. Working day movement model. In *Proc. 1st ACM Workshop on Mobility Models*, pp 33–40, Hong Kong, China, 2008.

[8] D. Huff. *How to Lie With Statistics*. W. W. Norton & Company, 1954.

[9] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *Proc. ACM MobiHoc 2008*, pp 241–250, Hong Kong, China, 2008.

[10] A. Keränen, J. Ott, and T. Kärkkäinen. The ONE simulator for DTN protocol evaluation. In *Proc. SIMUTools '09*, pp 1–10, Rome, Italy, 2009.

[11] L. Pelusi, A. Passarella, and M. Conti. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *IEEE Communications*, 44(11):134–141, 2006.

[12] B. Schneier. *Secrets and Lies: Digital Security in a Networked World*. Wiley, 1 edition, 2004.

[13] A. Shikfa, M. Onen, and R. Molva. Privacy in content-based opportunistic networks. In *Proc. 2nd IEEE Int'l Workshop on Opportunistic Networking*, pp 832–837, Bradford, UK, 2009.

[14] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. Tech. Rep. CS-200006, Duke University, 2000.