Amogh Pradeep*, Hira Javaid, Ryan Williams, Antoine Rault, David Choffnes, Stevens Le Blond, and Bryan Ford

Moby: A Blackout-Resistant Anonymity Network for Mobile Devices

Abstract: Internet blackouts are challenging environments for anonymity and censorship resistance. Existing popular anonymity networks (e.g., Freenet, I2P, Tor) rely on Internet connectivity to function, making them impracticable during such blackouts. In such a setting, mobile ad-hoc networks can provide connectivity, but prior communication protocols for ad-hoc networks are not designed for anonymity and attack resilience. We address this need by designing, implementing, and evaluating Moby, a blackout-resistant anonymity network for mobile devices. Moby provides end-to-end encryption, forward secrecy and sender-receiver anonymity. It features a bi-modal design of operation, using Internet connectivity when available and ad-hoc networks during blackouts. During periods of Internet connectivity, Moby functions as a regular messaging application and bootstraps information that is later used in the absence of Internet connectivity to achieve secure anonymous communications. Moby incorporates a model of trust based on users' contact lists, and a trust establishment protocol that mitigates flooding attacks. We perform an empirically informed simulation-based study based on cellphone traces of 268,596 users over the span of a week for a large cellular provider to determine Moby's feasibility and present our findings. Last, we implement and evaluate the Moby client as an Android app.

Keywords: anonymity; privacy; anonymous messaging; censorship; delay-tolerant networking; mobile networks

DOI 10.56553/popets-2022-0071
Received 2021-11-30; revised 2022-03-15; accepted 2022-03-16.

***Corresponding Author: Amogh Pradeep:** Northeastern University & EPFL, E-mail: amoghbl1@ccs.neu.edu
Hira Javaid: Northeastern University, E-mail: hira@ccs.neu.edu
Ryan Williams: Northeastern University, E-mail: williams.ry@husky.neu.edu
Antoine Rault: EPFL, E-mail: antoine.rault@epfl.ch
David Choffnes: Northeastern University, E-mail: choffnes@ccs.neu.edu
Stevens Le Blond: EPFL, E-mail: stevens.leblond@epfl.ch
Bryan Ford: EPFL, E-mail: bryan.ford@epfl.ch

1 Introduction and Motivation

Internet blackouts have affected numerous countries around the world. The causes of these blackouts vary: natural disasters in Taiwan [30], cyber attacks in Germany [31] and Liberia [37], and censorship cases such as Ethiopia [6], Iraq [38], Syria [8], and India (in certain regions) [20]. In this paper, we consider blackouts caused by active adversaries who seek to disrupt freedom of communication by attacking traditional forms of network connectivity; such adversaries also attack alternate (*e.g.,* ad hoc) communication infrastructure used during blackouts. The goal of this work is to provide secure, DoS-attack-resistant communication between users in blackout-affected regions.

Current metadata-private communication systems [14, 41] rely on the presence of a wide-area communication channel, such as the Internet, to provide secure communications. However, these systems are rendered ineffective in environments where Internet outages occur, as they rely on a wide-area channel to function. Previous attempts at providing some form of communications during blackouts include Rangzen [28], Briar [11], and FireChat [32]. Rangzen provides a sender-anonymous micro news service while Briar provides secure messaging capabilities. Both Rangzen and Briar require that users physically meet to exchange information required to use the services, which is time consuming and has typically failed in previous systems that use this mechanism (*e.g.,* PGP's web-of-trust [44]). Firechat has officially stated that “it was not meant for secure or private communications" [45]. Thus, there remains a need for a practical, secure communication system that works in blackout environments.

We present Moby, a blackout-resistant anonymity network for message communication via mobile devices. Moby presents a novel protocol that provides end-to-end encrypted messaging with forward secrecy and sender-receiver anonymity. Moby uses a bi-modal design consisting of a wide-area channel and an ad-hoc communication channel that are used in combination to provide blackout-resistant communications. The wide-area, Internet channel bootstraps the information necessary for communication via the ad-hoc channel. As we expect both wide-area and ad-hoc networks to be attacked during blackouts, the Moby network protocol provides resistance against network attacks. Specifically, the Moby network protocol augments an Epidemic routing [42] protocol by incorporating trust to provide DoS attack resilience. Each client in the Moby network computes trust for other participants of the network based on direct communication and mutual communication partners, and the client uses this information to make message queuing and routing decisions. These two channels combine to provide sender-receiver anonymous, end-to-end encrypted, forward-secret messaging to its users.

To measure the effectiveness of Moby's trust-based routing protocol, in terms of message delivery rates and message latencies, we perform an empirically informed simulation-based study using anonymized call data records from a large cellular provider's network, comprising 268,596 users in a large metropolitan area for a typical work week.[1] We run numerous simulations while varying environmental settings to determine the effectiveness of our protocol in different scenarios. In the absence of an adversary, Moby's, Epidemic, and Firechat routing protocols each achieve 50.73% delivery ratio; however in the face of a DoS attack, the delivery ratio for Epidemic/Firechat routing drops to 1.15%, while Moby achieves 13.96%.

Finally, we present an implementation of the Moby client as a modification of the Signal messaging application. Moby can run on any messaging platform that provides a secure channel; we chose Signal because it is open source and widely used. To determine the feasibility of our client on mobile devices, we perform power and processor utilization measurements.

Our main contributions are as follows:

– We present the design of Moby, the first blackout-resistant network for message communication for mobile devices using a bi-modal operation model.
– We present a trust establishment mechanism that defends Moby against denial-of-service attacks.
– We provide a trace-driven evaluation of Moby's performance and robustness with large real-world data.
– We present an open-source implementation of Moby and its performance in terms of energy consumption.

1 Use of such data was approved by our IRB.

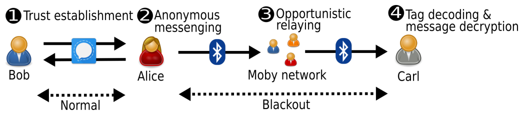

Fig. 1. Overview of Moby's bi-modal operation.

2 Goals and Assumptions

2.1 Goals and Non-Goal

Best-Effort Communication During a Blackout: Moby aims to provide communication via short messages during *blackouts*, which we define to be periods when wide-area Internet connectivity is unavailable (*i.e.,* IP messages are not routed). Moby uses short-range link layer technologies and promises best-effort delivery of its network messages.
Secure, Anonymous and Decentralized: Moby aims to provide confidential communication between a pair of users while maintaining message integrity and providing sender authentication. It also aims to make communications forward-secret and anonymous in the sense that, using a message, the associated sender or receiver of that message cannot be identified. It aims to provide these features without relying on a central authority.
Attack-Resilient Communication: Moby aims to provide attack-resilient communication during a blackout. We consider an adversary whose goal is to disrupt the Moby network (*e.g.,* a DoS attacks, jamming, etc.).
Usability: Moby is designed to enable communication for relatively nontechnical people. The Moby client is implemented as an extension to an existing, popular, secure messaging app to facilitate ease of use. With the extension, the app supports both communication over the Internet and via the Moby network .
Not Steganographic: Moby does not hide users connected to its network. Hence, it is trivial to identify network participants at a geographic location.

2.2 Assumptions

Secure Channel: We assume the existence of a secure wide-area communication channel at some point before a blackout. This channel ensures confidentiality, integrity, replay protection, as well as mutual authentication between users. In our implementation, we use a secure messaging application (Signal) run over the Internet to provide this feature. This channel is used to bootstrap information required for clients to function in the absence of the wide-area communication medium.

Trust in Contacts: The Moby design uses a notion of trust built on users' contacts. We assume that a user trusts contacts they communicate with frequently. In our simulations, we take this to be the set of all users stored as a contact in a client's phonebook who have been contacted more than a threshold number of times[2], under the assumption that such contacts are real people and known to users (*i.e.,* trusted at some level). In practice, users can customize their lists of trusted contacts to augment this heuristic with knowledge of who is trustworthy. We further develop this notion of trust to include pairs of users who have never directly communicated, but who share a large portion of mutual contacts.

2.3 Threat Model

We present Moby's threat model in this section using terminology introduced by Diaz *et al.* [17] and later (§ 4) show how we defend against it.

Inducing Blackouts: The adversary is global and active for wide-area communication channels (*e.g.,* IP-based communication via cellular or WiFi networks). Specifically, the adversary can induce blackouts of the secure wide-area communication channel that is used for communication in the absence of a blackout. These blackouts could span a city, state, or even a country.

The adversary is local for the ad-hoc Moby network. A global adversary for Moby network is impractical, as it requires covering the entire geographic region with short range radios. Thus, we omit such global attacks from our threat model. Below we consider the active and passive attacks this local adversary can mount.

Passive Monitoring: The local adversary can be passive and monitor a fraction of all Moby network traffic.

Introducing Malicious Messages: The local adversary can actively introduce arbitrary messages into the Moby network to attack the network. These messages can be introduced at arbitrary locations to attempt denial-of-service attacks on the network. Further, the adversary can modify messages it observes and send those modified messages into the network.

Active Jamming: The local adversary can actively jam network connections. Recall that jamming the entire network is not possible because the adversary is local. Further, such global jamming would be impractical given the geographic span of the network proposed.

User Coercion: The adversary can mount internal attacks, but these are by definition local. In this case, the adversary is active and launches network attacks from existing members of the Moby networkṪhe adversary can also launch passive monitoring attacks from such compromised hosts, contributing a vantage point to the passive monitoring mentioned above.

3 System Design

The Moby ecosystem is composed of several components that combine to provide blackout-resistant communication. Fig. 1 depicts the sequential operation of Moby. The trust establishment protocol is performed over a secure wide-area communication medium (step 1 in Fig. 1). In the case of a blackout (steps 2-4 in Fig. 1), each Moby client participates in the Moby network (step 3 in Fig. 1) by relaying Moby messages. A client can introduce messages into the Moby network (step 2 in Fig. 1) and receive messages from it (step 4 in Fig. 1). We refer to this property of using two communication channels as bi-modal operation. A client can simultaneously participate in both these modes.

A Moby client, via the secure wide-area communication channel, establishes trust with other clients. Our implementation of Moby does this with every contact on its phonebook that also runs Moby. Clients update their list of trusted clients based on the outcome of this step, and use this list in the Moby network to decide whether to hold or drop messages they receive.

The Moby network is a blackout-resistant anonymity network. We expect Moby users to install and use Moby as their default messaging application. Moby clients utilize this network to originate, receive, and relay Moby messages within it. During periods of Internet connectivity, the Moby network ad-hoc network can be used to provide sender-receiver anonymity, a property absent from many widely used Internet messaging applications. Moby builds *trust scores* between communicating users as they exchange messages with each other. During periods of Internet blackouts, the Moby network is the only medium through which clients can communicate. Moby provides best-effort delivery during these blackouts and uses *trust scores* computed earlier to prioritize messages.

2 We use a threshold of one to get an *upper bound* on the impact of trust in our simulation results, but this can be a user-configurable parameter.

3.1 Trust Establishment Protocol

The trust establishment protocol is performed by a pair of Moby clients over a secure channel. Moby clients bootstrap information for each other using this protocol. To participate in the Moby network, clients use a list of trusted clients (referred to as a *Trust List*) to decide whether a Moby message should be held or dropped when received. Clients check this list to see if they recognize the client that relayed the message, to make that decision (§3.2.6). Clients use cryptographic material that provides forward-secret, out-of-order resistant, end-to-end encrypted messaging. A pair of clients use this material to encrypt messages sent via the Moby network. To satisfy these properties in our implementation, we use the Double Ratchet (DR) algorithm [40]. Signal's [2] implementation of DR includes sender and receiver identifiers in messages, we do *not* include these identifiers in Moby to ensure sender-receiver anonymity (further discussed in §3.1.2). A client uses a long-lived signing key and a corresponding verification key to recognize clients in the Moby network (§3.2.4).

At the end of trust establishment both clients will have: updated Trust Lists, cryptographic materials for communication, and verification keys for each other.

The trust establishment protocol must run over a pre-existing secure communication channel, regardless of the medium of communication. Thus, it can run either over the Internet, given an underlying channel (*e.g.,* a messaging application) that is secure, or manually when two clients meet each other. Note, however, that the Moby network should not be used as this secure channel. For a given pair of clients to communicate via the Moby network, the trust establishment protocol must be run at least once between them. In our implementation, this protocol is run against all contacts in a client's phone book on first installation of the application and whenever a new contact is added on a given device. It could be run more or less frequently, depending on the deployment scenario. Trust scores computed do not change during a blackout and should only be updated once a secure channel for trust establishment is restored.

We now present each component in detail and how they are used in Moby.

3.1.1 Trust in Moby

The concept of trust in Moby is realized by message trust values and user trust lists.

Trust Value Calculation and Updates: A Moby client computes the trust value associated to another client on completing a *Moby Handshake*. Clients use information about how many contacts they share in common and how many times an opposite client is contacted to calculate the trust value for that client. We describe a general approach for computing trust, then provide details of the specific formulation of trust value computation that we use in our system.

To compute a numeric trust value for a pair of clients, we consider the following abstract function:

$$\text{Trust Value}_{\text{client}} = f\left(contacts,\ communications\right) \quad (1)$$

To extend a client's trust list, we support the notion of indirect trust that incorporates the notion of transitivity: if client A trusts client B, and client B trusts client C, then client A trusts client C. In this example, A and C are 1-hop trusted clients (assuming there is no direct communication between A and C to otherwise establish direct trust).

Each Moby message contains a trust value in it as well. This value is first set by the sender and is updated at each client that receives it and relays it in the Moby network. Clients update this value based on who forwards the message. Thus, for a message:

$$\text{Trust Value}_{\text{message}} = f\left(old\ value,\ client\ value\right) \quad (2)$$

In our implementation, we use binary trust: all contacts are marked either trusted or untrusted. Similarly, all messages are marked trusted or untrusted. We use binary trust as it captures the upper bound that can be achieved by other trust models for a given hop count. A fine grained trust system could perform, at best, as well as our binary model. These update operations therefore tell us whether an opposite client is trusted and likewise whether a message is trusted.

The binary trust system is one method of computing trust values. It provides a coarse-grained model of trust where entities are either trusted or untrusted. Computation of trust values are configurable; a real number trust value that provides a fine-grained model of trust could be used.

Trust Lists: A trust list is comprised of the following information for each client that is trusted: unique contact information, a verification key, a trust value, and a hop value. The contact information is a unique identifier for that client, e.g. username or phone number. The verification key is a public key for the signing key used by that client. The trust value is a numerical indicating how much that client is trusted. The hop value is the shortest distance to the client (e.g. if client A trusts B, and B trusts C, then A has a hop count of 1 for C).

Each Moby client maintains an instance of a Trust List; while participating in the Moby network, this list is used to check whether an encountered client can be trusted and to what extent. These trust lists are populated and updated during *Moby Handshakes.*

Trust lists could potentially be polluted with stale contacts (*i.e.,* ones in contact lists that are no longer trusted). One mitigation is for Moby to use only contacts that have been contacted recently. Setting proper thresholds on recency, however, remains an open question. As an example value, our evaluation assumes trust only for contacts that have been communicated with over the past year.

3.1.2 Cryptographic Material

To communicate via the Moby network, a pair of Moby clients must share some cryptographic material. In our implementation, we use the Double Ratchet (DR) algorithm. DR is used with just the symmetric key ratchet option which provides forward-secret, out-of-order resistant end-to-end encryption. We drop the second ratchet step as it provides break-in resistance (which is outside of our threat model) and requires messages between clients to be reliably delivered (which the Moby network cannot provide). To gain sender-receiver anonymity, we drop all identifiers related to senders or receivers (we provide a heuristic proof for this in Appendix A. By doing so, it becomes computationally more expensive to check message reception, but we are willing to trade computation for security here. Although we use DR in our implementation, any cipher that provides the same guarantees could be used.

3.1.3 Moby Handshake

The *Moby Handshake* is a four-message protocol performed by two Moby clients. We use the PSI-CA [15] protocol, and exchange: verification keys, cryptographic material, and trust lists, and establish trust between two clients. We refer to the client that starts the protocol as the *initiator*, and the opposite client the *receiver.*

Private Set Intersection Cardinality: PSI-CA is an asymmetric 2-message protocol that computes, for the *initiator*, the cardinality of set intersection for two sets, while keeping the contents of the sets hidden from either party. A pair of Moby clients use PSI-CA twice in the handshake to compute the overlap of trusted contacts (using verification key fingerprints as input to the protocol) without revealing the fingerprints.

Handshake Steps: Fig. 2 illustrates the PSI-CA handshake steps. The *initiator* starts the PSI-CA protocol with the *Hello* message and receives a *Hello Response* from the *receiver.* This concludes one round of PSI-CA

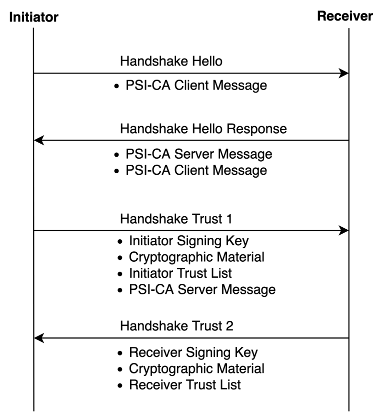

Fig. 2. Messages in one round of the Moby handshake.

and starts the second round. Using the *receiver*'s message, the *initiator* computes the overlap of trusted contacts it has with the *receiver.* The *initiator* decides to trust the *receiver* if this overlap is above a threshold. If the *receiver* is not trusted, the *initiator* aborts the protocol; if the *receiver* is trusted, the *initiator* responds with its verification key, some cryptographic material, its trust list, and the PSI-CA message required by the *receiver* as per the PSI-CA protocol(*Trust 1* in Fig. 2). The *receiver* can now compute the overlap it has with the *initiator*, ending round two of PSI-CA. If the overlap is above a threshold, the *receiver* responds with its own verification key, some cryptographic material, and its trust list (*Trust 2* in Fig. 2). At the conclusion of Trust 2, the *Moby Handshake* protocol is complete.

At the end of a successful Moby handshake, both *initiator* and *receiver* process the trust lists they received. Each entity computes a trust score for the opposite entity (as described in §3.1.1) and adds it as a direct trusted contact (0 hop value). They then increment the hop count of each element in the received trust list (to signify that the trusted entity is one extra hop away) and add them to their own trust list. If an entry already exists, it is added only if the hop count is lower than the previous value.

3.2 Moby Network Protocol

We now present the *Moby network Protocol*, which uses trust lists, message exchanges, and post-exchange protocols to enable clients to propagate messages while resisting denial-of-service attacks.

The Moby network consists of a set of Moby client's spread over a geographic region, using the Moby routing protocol. The Moby network uses an augmented Epidemic [42] routing protocol; when a pair of clients encounter each other, they exchange all the messages they currently store. We improve this classical protocol with Trust Lists, which determine whether a client stores or drops messages. Last, we define an optional post exchange protocol for cases where a pair of clients do not recognize each other.

3.2.1 Link Layer Technologies

Our protocol uses wireless network technologies to detect whether a participating client is in communication range and to exchange messages with these clients. In the case of our implementation, the ones used are Bluetooth and Wi-Fi Direct. We use these link layer protocols because they are widely supported on mobile devices, but any point-to-point technology could be used.

3.2.2 Client Discovery

The *Moby Client Discovery* step involves discovering other clients that are within the communication range of the client executing it. If multiple such clients exist, *Discovery* returns a list of all of them and each one of them is used in the next set of steps. Moby is not designed to hide the fact that it is installed on a device, thus, we do not require this link layer protocol to satisfy any security properties. In our prototype, we use Wi-Fi Direct for usability reasons (see §6).

3.2.3 Sending and Receiving Moby Messages

Moby Network Message: In the context of the Moby network, a Moby message contains the following components:

– A plaintext Time to Live (TTL) value, set by the *sender* of the message.
– A plaintext Trust value, initially set by the *sender*, changed at each client that relays the message (see §3.2.4).
– An end-to-end encrypted payload and associated *MAC*, using the ephemeral keys that the *sender* and *receiver* share.

The TTL is a (wall-clock) timestamp after which the message should be dropped by all clients. We require the TTL and Trust values to be in plaintext as they need to be read/modified by all Moby clients; attacks on these values are discussed in §4.3. A *sender* must first execute *Trust Establishment* (§3.1) with the *receiver* to send them a message.

Sending a Message: The *sender* produces an encrypted payload and *MAC* via the Encrypt-then-MAC method using the shared cryptographic material. It then sets a large trust value and a random large TTL value binned by the hour and places the message onto its own Message Queue. (A random, binned TTL value prevents a local attacker from tracing message origin to the sender, to an extent.) Messages propagate the network when *Message Exchanges* occur.

Receiving a Message: When new messages are added to a client's Message Queue following Signature verification and trust value updates, the messages need to be checked to see if the intended destination is the receiving client. A receiving client checks each new message with each session it has for its contacts. If, for any session, a computed *MAC* matches the *MAC* attached to the message, the receiver knows the sender of the message and can decrypt it. We present pseudocode for this in Appendix B. Note each message remains in the receiver's queue until the queue policy (§3.2.6) determines it should be dropped. This prevents an adversary from inferring client message reception.

3.2.4 Message Exchange

Moby Message Exchange involves sending a Message Queue, a *Salted Fingerprint*, the salt used, and the *Signature* generated over both the queue and fingerprint. Clients use the *Client Discovery* protocol to identify hosts to perform Message Exchanges with. If multiple clients are discovered, a random client is chosen to perform this step.

Message Queue: The Moby Message Queue is an ordered list of Moby messages sorted based on the message trust values. As we use binary trust in our evaluation, all trusted messages would be at the front of the queue with untrusted ones behind them.

Signature: The Signature is produced by a client that sends the queue. The key used to produce this Signature is the signing key corresponding to the verification key advertised by the client during *Trust Establishment* (§3.1). Given this verification key, the receiver of a queue can verify the integrity of the queue and infer the identity of the forwarder. The protocol uses this information to decide how to process the new messages received.

Salted Fingerprint: Signature verification is an expensive process as compared to hash computation; knowing the verification key for the Signature would make this more efficient. Adding a key fingerprint would achieve this; we make sure to use a salted hash of the fingerprint constructed as follows $\mathsf{H}(\textit{fingerprint}\ \|\text{salt})$ where salt is a random string of at least 32 characters. We salt this fingerprint to prevent adversaries from observing fingerprints they have not seen before. Thus, a client can identify which verification key to use by first computing the hash of the salt with all stored fingerprints and comparing it with the received hash. Following this, the client verifies the Signature, and can associate the message to a sender (in case a match is found). We salt fingerprints to protect verification keys in the network from adversaries (§4.3).

3.2.5 Post Exchange PSI-CA

To compute trust scores of clients not encountered before a blackout, a pair of Moby clients can perform a PSI-CA computation following a successful Message Exchange. This step is optional as PSI-CA is an expensive operation, both in terms of computation and communication overheads.

In our implementation, we always perform this step, which involves exchanging PSI-CA protocol messages between a pair of clients. The input sets are populated using *fingerprints* of signing keys of already-trusted clients. Based on the output of this protocol, a client decides to trust or not trust the opposite client. No other information is to be sent in this step as the underlying link layer technologies provide no security guarantees.

3.2.6 Message Queue Policy

Moby clients have limited resources in terms of storage space and bandwidth for message exchange. Thus a message queue policy is necessary for a client to decide whether to hold or drop messages.

When a Moby client receives new messages, it updates each message's trust in the queue based on the client from whom it is received. Following this, new messages are added to the local queue based on this updated trust value. Message queues are sorted lists based on trust values. In case the message queue fills up, messages with the least trust value are dropped. If multiple least-trusted messages have the same trust value, one of them is chosen at random and dropped. This prioritization of trust values to store messages helps prevent network congestion attacks and makes sure that legitimate messages stand a chance of delivery when adversaries flood the network with dummy messages. A flood would result in queues filling up, but legitimate messages would be at the front of these queues given that they should on average have higher trust values compared to messages from adversaries.

In the binary model of trust values, the implications of this policy would be that trusted messages are always prioritized over untrusted ones. If the queue is filled with trusted messages, one would be dropped at random. Similarly, untrusted messages are picked at random when they need to be dropped.

3.2.7 Moby Data Structures

A client that participates in the Moby network maintains the following data structures for forwarding messages: a signing key, a Trust List containing contacts' verification keys and cryptographic state for each direct contact, and an ordered list of Moby messages.

The signing key is an asymmetric key, with an associated verification key and fingerprint. These are used to sign message queues and verify signatures as shown in §3.2.4. The message queue contains all Moby messages the client wishes to send or relay sorted by the trust value associated to that message.

The list of cryptographic states associated to users are established in the Moby Handshake (§3.1.3) and enable this client to communicate via Moby messages. Using this cryptographic state, a client must be able to encrypt a message and compute a *MAC* for it or verify a *MAC* and decrypt the corresponding message.

4 Security

In this section, we discuss a variety of attacks and how the Moby ecosystem defends against such attacks. The adversary's capabilities are described in the threat model (§2.3). We briefly discuss the security of the wide-area communication medium and then discuss attacks on the Moby network.

4.1 Wide-Area Communication Medium

We assume there is a wide-area communication medium (e.g., the Internet) that can be disconnected by the adversary. We further assume that secure messaging over the wide-area medium exists (e.g., end-to-end encryption via Signal); securing the wide-area channel is considered orthogonal to Moby's goals.

4.2 Denial-of-Service (DoS) Attacks

We now discuss several Denial-of-Service attacks on the Moby network.

Local Jamming: Using specialized jamming hardware that blocks the link-layer communication medium, the adversary could prevent Moby network communications within a small geographic region. Given the geographic span of the Moby network, the adversary could perform this attack on a part of the network; we present our findings related to such an attack in § 5.5.2. No known defenses exist for such jamming techniques; users in the jammed region would be directly affected, but those outside the region would not.

Network Flooding: An adversary could perform a denial-of-service attack on the Moby network by flooding the network with malicious messages sent at arbitrary locations. Doing so, the adversary would try to fill the queues of all participating clients; legitimate messages would be dropped by clients as their queues would be filled with malicious messages.

Moby uses trust values to defend against such attacks. A malicious message would gain only as much priority as the client that introduces it into the network.

We argue that adversaries will find it difficult to obtain high trust values. The main reason is that Moby primarily incorporates trust based on contacts in a user's phonebook, something that is difficult for an adversary to control. Even if an adversary were to convince a user to add malicious contacts to a phonebook, we argue that either these would be small in number relative a user's non-malicious contacts, or there would be so many malicious contacts that it would be conspicuous and detectable. Thus, we assume that on average malicious clients will have substantially lower trust values than non-malicious ones.

Given that malicious clients would not have high trust values in the legitimate network, these messages would get discarded faster than legitimate messages. We show the resilience of Moby to such an attack in §5.5.3.

User Flooding: In such an attack, an adversary sends a number of Moby messages to one specific Moby client. The goal of this is to exhaust the resources of that client, by causing it to perform operations on this message (storing, processing, etc.). Such an attack has limited effectiveness. For an adversary to send a message to a client, trust establishment needs to be performed with that client first. Therefore, this can only be performed by a client's trusted entities and not all clients.

Flooding via Coercion: In a coerced flooding attack, the nodes that flood the Moby network with malicious messages lie within the system. These attackers would be on some contact lists and use this trust to carry out an attack. Such attacks are effective, but limited given that a small number of users can be coerced. We show the extent of such attacks in §5.5.4

Handshake Attack: An adversary could initiate many handshakes with clients within wireless range, to get them to waste energy performing computations. Such an attack would not scale as multiple malicious nodes would need to be present at a specific location to attack all users. If there are few such attackers, legitimate nodes in the area would still be able to perform exchanges; if there are many attackers, this attack becomes a jamming attack (discussed earlier). Thus, this attack could disable relatively small geographic regions of the network where the attackers are located, but it would not disrupt the entire Moby network.

4.3 Attacking a Moby Message

These attacks involve trying to exploit a Moby message to obtain information about communicating clients or to disrupt its propagation through the Moby network.

Sender/Receiver Tag Identification or Spoofing: The Moby MAC as described in 3.2.7 has the property that it can be verified only if a user has knowledge of a secret. This secret is known only to the sender and receiver. Thus, an adversary cannot identify the sender or receiver given the MAC without having compromised the shared secret. Further, they cannot produce a MAC that is accepted by a receiver and hence cannot spoof the sender or receiver for a given message. Lastly, messages are introduced and removed from message queues of clients in a way that prevents local adversaries from inferring the senders and receivers of messages (as discussed in §3.2.3).

Attacking the TTL Value: Moby messages contain TTL values that are in plaintext (can be read/modified by any intermediary client). Tampering with the TTL value is possible and would be equivalent to introducing a malicious message into the network. These modified messages would only gain as much priority as the entity that modified it. Thus, regardless of the TTL values, these modified messages would get discarded sooner than legitimate messages based on trust values. Attacking message TTL values would therefore not affect the overall performance of the Moby network.

Payload Attack: An adversary could try to obtain information about the message being sent between a pair of users based on the payload of a Moby network. To prevent this, all message payloads are end-to-end encrypted; only the sender and receiver of the message can read the contents of the payload, given the adversary cannot break standard cryptographic building blocks.

Fingerprint Enumeration: Fingerprints associated to a clients verification key are used as input in the PSI-CA protocol (in both places PSI-CA is used). Thus, if an adversary has knowledge of all the verification keys used in the Moby network it could use that in Equation 1 to maximize the first component. Moby protects against this using salted fingerprints so enumeration is not possible.

4.4 Moby Network User Inference

Moby uses link-layer technologies that broadcast the use of Moby. Thus, it is easy to infer that a person is running Moby client on their device based on these broadcasts. We do not defend against such attacks.

4.5 Trust Link Identification

The adversary could try to infer which user trusts which other user in the Moby network in this attack. This would be possible if the adversary could infer trusted contacts in the handshake phase. We use PSI-CA to protect against this; an entity that performs the handshake only obtains the cardinality of overlap and not which elements overlap itself. Further, tricking an entity into performing a handshake with the adversary is a challenge as well, which protects against such identification attacks.

4.6 Trust Establishment Attack

An adversary could try to maximize its trust value for other attacks on the Moby network. This is prevented as follows. The trust establishment step is only performed with users that are sufficiently trusted (§3.1). Further, the overlap value computed using PSI-CA is capped; an upper bound is set to the number of elements used as input in that protocol. By setting reasonably conservative limits, one can mitigate dictionary attacks on client trust lists (*i.e.,* when an attacker claims to trust all Moby clients to maximize set intersection values).

4.7 Post Exchange PSI-CA Attacks

The PSI-CA exchange, when done over an insecure channel could leak some information. The fact that a pair of clients are performing this exchange would imply that they do not trust each other. On the other hand, if this is executed even if clients trust each other, it would add unnecessary overheads to the Moby network. Thus, this piece of the system is marked optional and is up to the client and/or implementer to decide.

Next, a man-in-the-middle attack could be performed if PSI-CA were performed over an unreliable channel. Although authenticated PSI exists [16], it is not sufficiently fast for cardinality computation and thus we do not use it. As securing the PSI-CA protocol over an insecure channel is not a goal of this paper, we do not explore this any further.

4.8 Out-of-Scope Attacks

Targeted Attacks: Attacks where one user, or a set of users are targeted by the adversary whereby they are followed, monitored, burglarized, coerced, and the like. Such attacks can not be solved with a technical solution.
Secure Devices: We do not defend against attacks on Moby client devices via malware, or other side-channels.

5 Simulation-Based Evaluation

To evaluate Moby's system design at scale, we perform trace-based simulations on our filtered dataset containing cell tower data of 268,596 users for the span of a week. We begin by describing our data, simulation framework, and simulation experiments. We then use our experiments to analyze the impact of key Moby parameters and the simulation environment on our key performance metrics: message delivery ratios (number of messages delivered to number of messages sent) and message latencies. We evaluate Moby both in the absence of an attack, and using an adversary who floods the network with dummy messages as part of a DoS attack. We find that Moby substantially outperforms Epidemic/Firechat under a network adversary. Firechat uses Epidemic routing over a mesh network and thus is identical to the Epidemic routing we evaluated. In the best-case scenarios we investigated, Epidemic/Firechat routing achieves 1.15% delivery rate under an adversary, while Moby achieves 13.96% in the same conditions.

Note that Moby uses contact lists, call histories and hop counts to realize trust graphs while Rangzen [28] uses an already provided social graph. Thus we could not compare these approaches due to a lack of a large mobility dataset with both call information and social graphs. We could not compare with Briar [11] because it lacks a formal specification. Finally, we investigated using other mobility traces (*e.g.,* from taxis) but could not identify a realistic way to map those mobile nodes to Moby trust graphs.

5.1 Dataset

Our dataset contains call data records from a large European cellular provider's network deployment, gathered in 2009. During that one year, we observe that 25,719,853 users placed 6,000,444,782 phone calls and sent 1,642,489,960 messages. The dataset specifies the cell tower (and its geographic location) used for each phone call or text message, which allows us to roughly geolocate a user at that moment of communication. Each user is anonymized and assigned a randomly generated unique ID; thus our data contains no user identifiers whatsoever. (Note that the dataset contains neither user-identifiable data—phone numbers are replaced with random strings—nor precise geolocations, the data is kept in secure access-controlled environments, and the research protocol was approved by our IRB.) We discuss the limitations of our dataset in §7.

Data Filtering: We perform a number of filtering operations on this dataset to use in our simulation framework. First, we filter the data to consider a specific geographic region; the span of the region is that of a highly populated city in the European country for which we have data. Next, we analyze the pattern of communications that users in that region have for the span of the year of data. We do so to pick the most representative week of the year on which to run our simulations. We compare communication statistics in terms of calls placed, messages sent and number of users that participate for a given day. Lastly, we filter users that send/receive messages in our simulation by a liveness metric which is the number of hours a user is observed communicating (via calls or messages) in the span of days we select. After the filtering steps, we end up with 268,596 users (liveness of 1) who are simulated. For plots in this section, we use a liveness of 4 (78,486 users).

Tower Distribution: After filtering, our simulations use 786 towers that cover an area of $\approx 180km^2$. To approximate range of the cells corresponding to these towers, we calculate the average distance between each tower and its five closest neighbors. Averaging this value across all towers in the network and dividing by two to get a radius, we find the the average range of towers to be $118m$ (σ =86m). Thus, for much of the region, the range of cells that users connect to is relatively small, and thus users connected to the same cell are in many cases capable of connection via the wireless channels needed for the ad-hoc network.

5.2 Simulation Framework

We implement a custom network simulator to measure the performance of Moby as well as previous systems. The simulator uses call data records to simulate users and their movements, and advances time in one-hour increments, due to the one-hour granularity of our dataset. Each user in the system has a message queue; messages sent by the user are added to the queues at the right hour (based on when they are sent). When a pair of users are sufficiently close to each other (associated to the same cell tower for an hour), a message exchange is simulated between them. At message exchange time, the simulator executes the Moby routing using trust information for the corresponding pair of users. Last, the simulator can simulate network adversaries that perform attacks on the network.

Thus, we have a framework to simulate the Moby network and measure the guarantees it provides if deployed in the real world. Using the simulator we obtain performance metrics in terms of message delivery ratios and message latencies.

5.3 Experiments Performed

To compare Moby with previous systems for ad-hoc networks, we perform network simulations using the Epidemic routing algorithm, using our filtered dataset. We explore a number of different simulation and Moby parameters, including some that are relevant only for attack scenarios (*e.g.,* volume of attack messages injected into the network).

Note that in the absence of an active network adversary, we found that the Moby routing protocol performs nearly identically to Epidemic/Firechat routing, and we show only Moby performance in those scenarios. We directly compare Moby and Epidemic/Firechat routing only under attack scenarios.

5.4 Simulation Parameters

This section details our simulation parameters (see Appendix C for a compact summary) and presents the values we explore along with why these were chosen.

Fixed Parameters: We begin by picking a geographic region corresponding to a large metropolitan area to filter out the set of users we simulate from the large dataset we possess. Moby is intended for deployment in dense urban areas that provide many opportunities for clients to exchange messages, so we do not evaluate suburban or rural areas. We use 3 days of data to drive our simulations after testing longer timescales (up to 7 days), as longer spans did not further improve delivery ratios, and results overall were not substantially different. We analyze the daily mobility and daily communication patterns for the chosen region for the year in terms of total calls made, total messages sent, and total users observed. We use this information to pick a set of days that match the average number of messages sent in a region over 3 days; the days chosen are the 53rd to the 56th day of the year.

Based on the communication patterns of the participating users, we set the number of messages to be sent to 30,000. Messages are sent in the first 48 hours of the simulation with “cooldown" period of 24 hours when no new messages are introduced to the network, to allow time for delivery of recently generated messages. Sources for these messages are picked randomly from the set of users, and corresponding destinations are picked based on the contact list of the source user. To account for any bias from source selection, we perform multiple simulations for each configuration, each with a different random selection of senders. We present average metrics since the range of results across such simulations was only 0.2–0.5%. To model a real-world message load on the Moby network, the number of messages sent each hour is proportionate to the text messages that users sent during that hour in our trace data. (We found similar results when using number of calls instead of text messages.) The contact list is set to a constant value for all simulations, details about this list are in §5.1.

Connectivity: Due to the lack of precise user location information in our dataset, we varied the percentage of users that perform a message exchange within a tower. We noticed a negligible drop in delivery ratios (under 1% for all TTLs) when half the pairs of users perform message exchanges; further, even when only 10% of user pairs connected to the same tower can exchange messages, performance of the network drops by less than 5% for TTLs 24 and higher and even lower (under 2%) for TTLs 48 and higher. To focus our analysis on other factors that affect the Moby network, our evaluations below use the setting where all users connected to the same tower per hour perform exchanges. For more details about performance with fewer users being able to conduct exchanges, see Appendix E.

Varied Simulation Parameters: We vary other parameters in their respective ranges to observe their effects and to monitor their interact with each other. We define liveness of a user as the number of hours the user appears in the dataset for the given span of days. Liveness is varied between 1 and 12 in increments of 2. The number of DoS messages injected into the network per hour varies from 0 to 10, that number of messages is sent to all users at that location at the given hour.

Varied Moby Parameters: We investigate several Moby configuration parameters to understand their trade-offs with respect to performance. Per user queue size is varied between 5,000 and 30,000 in increments of 5,000. The time to live (TTL) for each message is varied between 12 and 72 hours in increments of 12 hours. In our figures, when we refer to “TTL N”, we mean “N hours after each message was sent.” Thus our figures refer to average performance for a given initial TTL, and not wall-clock time in the simulation. The trust list parameter is varied. Different trust lists are generated for different hop values, i.e. hop 0 implies that all contacts are trusted, 1 implies that all 1 hop contacts are trusted as well, and so on. We explored nonbinary trust based on call frequency (where trust in a client is the number of communications with that client divided by the total number of communications). While the resulting trust values were asymmetric, we found that results were very similar to (symmetric) binary trust (further discussed in§ 7). Thus we use only the simpler binary trust approach in our evaluation. We evaluate the impact of indirect trust by varying hop counts among 0 (no indirect trust), 1 and 2.

A summary of parameters is in Appendix C.

5.5 Performance Results

We now present performance results for Epidemic/Firechat routing and Moby routing in terms of message delivery rates and message latencies when run using our filtered dataset. We find that Epidemic/Firechat and the Moby routing protocols performs nearly identically in the absence of an adversary. We then compare Moby with Epidemic/Firechat under an active denial of service performing adversary.

5.5.1 Performance in the Absence of an Adversary

We now investigate the performance of Moby without an adversary. Our goal is to identify the impact of Moby parameters (TTL and queue size) on message delivery and latency. We find that in the simulated environment, there are performance trade-offs for these parameters, and we highlight combinations that are most effective.

Delivery Ratios: For simulations in the absence of an adversary, we vary liveness, user queue size, and time to live (TTL). We observe that for different liveness values, delivery ratios vary, but overall trends for queue size and TTL hold; liveness is directly proportional to delivery ratio. Thus, we present metrics for simulations with a liveness of 4 (78,486 users), all other liveness values showed us similar trends.

To investigate the effect of varying queue sizes and TTLs on delivery ratios, we plot delivery ratio on the y-axis and TTLs on the x-axis and draw lines linking simulations with same queue size values in Fig. 3. For queue sizes less than 30,000 (all messages in the network), we find that delivery ratios first rise up to a

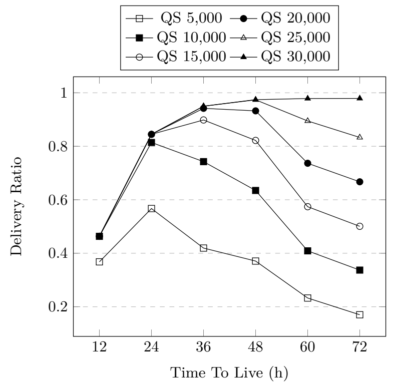

Fig. 3. Effects of varying time to live for a queue size in terms of delivery ratios. Increasing TTLs leads to a rise in delivery ratio up to a point after which a drop occurs due to filled user message queues.

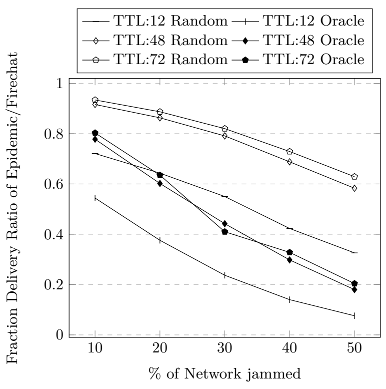

Fig. 4. Impact of jamming on delivery ratio compared to Epidemic/Firechat routing. Lower TTLs suffer worse delivery ratios under jamming, and jamming popular locations (Oracle) is more efficient than random, as expected.

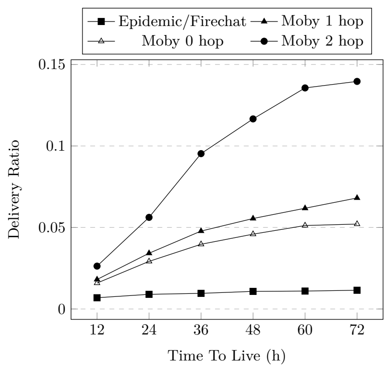

Fig. 5. Performance of Epidemic/Firechat, Moby 0, 1, and 2 hop during a DoS attack. Moby routing consistently outperforms Epidemic/Firechat in such environments, with higher trust hops performing better.

	Average Latencies (h)			
TTL (h)	**Minimum**	**Mean**	**Maximum**	**Avg Std Dev**
12	4.84	5.49	5.77	3.56
24	9.15	12.08	14.7	7.35
36	9.67	15.11	18.68	9.28
48	9.79	17.57	23.81	11.68
60	9.79	16.6	20.63	12.74
72	7.02	12.74	17.21	10.06

Table 1. Statistics for average latencies for the set of simulations grouped by time to live. Average latencies initially increase with TTL, and drop under very large TTLs due to higher number of messages being delivered closer to when they were sent.

point after which a drop occurs. Follow the line with solid boxes (queue size 10,000), we find that the delivery ratio first rises going from TTL 12 to TTL 24 but then begins to drop after 24. This tells us that for a given queue size value, there is an ideal TTL value at which the network performs the best, increasing TTL beyond that point causes the queues of numerous clients to fill up with messages that cannot be delivered, thus leading to lower delivery ratios.

As we increase the queue size, keeping TTL constant, we find that delivery ratios increase up to a point, after which there is no improvement (at around 30,000 queue entries).

Message Latencies: We now look at the average message latencies for a simulation, averaging values over the set of all delivered messages. We find that the spread of latency values for messages delivered in each simulation is large, and thus computed the minimum, mean, maximum, and standard deviation of latencies, averaged over the set of simulations with the same TTL; the results are in Table 1. Following the means (2nd column), we notice an increase until TTL 48, after which latency drops. We analyzed message delivery patterns and message latency patterns for each simulation, and found that most of the messages that get delivered in these simulations, get delivered close to when they were sent, but continue to occupy user queue space due to the TTLs being large. (Note that unlike in a traditional IP routing network, messages stay in the queues until the TTL expires.) This adversely reduces delivery ratios, but the average latency is lower because the messages that are delivered tend to be delivered quickly.

Setting Optimal Parameters: Most messaging systems strive for high delivery ratios and low latencies; we explore the effect of all parameters on both these statistics to find the optimal parameters to tune Moby. We primarily concentrate on the effects of both queue size and TTL on both delivery ratios and latencies. To summarize our analysis, we found that common factors for good performance are relatively large queue sizes (to increase delivery ratios) with moderately sized TTLs (36 hours or higher for larger queue size) to tame queue buildup. Under such conditions, Moby achieves high delivery ratios (>0.8) with latencies in the range of 15–17.5 hours. Please see Appendix D for more detail on our parameter space exploration.

5.5.2 Performance During a Jamming Attack

To understand the impact of a jamming attack on the Moby network we consider two attacks: randomly jamming locations and jamming the locations most visited by users (which we refer to as *oracle* attacks as it requires *a priori* knowledge). When a region is jammed,

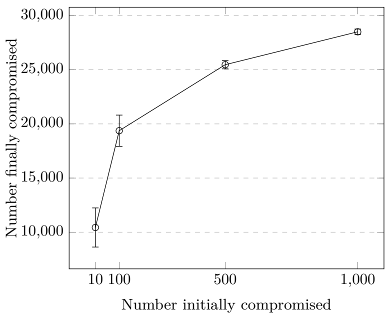

Fig. 6. To understand the extent of a compromised user attack, we vary the initial number of compromised users in the x-axis (10, 100, 500, 1000) and measure the total number of users that are affect at the end of the simulation in the y-axis.

no users can exchange messages in that region. The latter gives us an *upper bound* on the attack's effectiveness.

Fig. 4 plots the results of jamming attacks for different TTLs. The y-axis is the delivery ratios as fractions of the baseline result (Epidemic/Firechat without jamming), and the x-axis is the fraction of the simulated region that is jammed. Each line represents a TTL value and jamming strategy; we omit some TTLs because they exhibited similar trends as the ones shown. As expected, we observe that delivery ratios drop as the jammed region increases and random jamming is less effective than an oracle attack.

While these results may indicate that jamming can be an effective attack, it is important to put the results in context. Jamming just 10% of the Moby network involves covering a large geographic area (78 cell towers, $\approx 4.5km^2$) which already requires significant coverage and would likely incur substantial collateral damage. Further, obtaining a jamming oracle is difficult as movement patterns during a blackout would be unpredictable. Lastly, our results show that Moby is resistant to low amounts of jamming, for random or oracle based attacks. Thus, the effect of such an attack on our system is limited under practical constraints.

5.5.3 Performance During a DoS Attack

For simulations in the presence of an adversary, we vary liveness, user queue size, time to live, number of malicious messages sent, and the *contacted threshold* (*i.e.,* the minimum number of communications between users to be considered trusted, as explained in Table 2). We generate malicious messages as follows: we place an adversary at each message-exchange location (*i.e.,* cell tower) and send a fixed number of malicious messages to every user at the location every hour. In total, thousands of attack messages are generated per hour, while the number that get propagated depends on the number of users at the location at each hour. After testing various values for number of malicious messages sent per hour, we find that delivery ratios fall drastically (even for low rates). Based on this, we selected a rate of 10 messages per location per hour, as this was sufficient to make Epidemic routing's delivery ratio near zero.

We simulate Epidemic/Firechat routing in this malicious environment and compare the results we obtain to those for the Moby routing protocol. We plot a curve for each routing protocol (Epidemic/Firechat and Moby with different levels of indirect trust) with delivery ratios on the y-axis and time to live (TTL) values on the y-axis in Fig. 5. The solid squares line shows the performance of Epidemic/Firechat routing, while the hollow triangle, solid triangle, and solid circle lines show the performance of Moby routing with 0 (only direct trust), 1 (trust contacts of trusted contacts) and 2 hop trust metrics, respectively. Recall that our binary trust model provides an *upper bound* on message delivery when compared to other trust models.

Our first key observation is that Epidemic/Firechat performance is abysmal under attack—delivery ratios are approximately 0.01 across all tested TTL values. The reason is that the message queues are filled with malicious messages, leaving little-to-no room for legitimate messages to be forwarded or delivered. It also motivates the need for an attack mitigation strategy. Next, we note that even without indirect trust (the 0-hop curve), Moby substantially increases delivery ratios—by up to 5x in the case of large TTLs. We see a similar trend for larger hop values (allowing indirect trust), with the exception that larger hop values leads to substantially better delivery ratios (up to 0.14, an approximately 14x improvement over Epidemic/Firechat). This is because the network of directly trusted clients is quite sparse, and using indirect trust allows the network to be denser and thus support more trusted message exchanges during network attacks. Thus, while Moby cannot recover the original performance while under a large-scale attack, it still manages to deliver a substantial fraction of messages compared to the case of no active attack.

5.5.4 DoS via Compromised Users

We now estimate the extent of a DoS attack where an adversary uses trusted clients to flood and fill message queues of legitimate clients, thus leaving no trusted slots for legitimate communication. In this model, which is based on a simpler simulation that does not require modeling message transmission, a trusted user that comes in contact with a compromised user is considered to be DoSed. We focus on the number of compromised users instead of delivery ratios, as they represent the portion of the Moby network that is disabled independent of the message transmission patterns.

We pick initial users at random, varying the number picked in the x-axis, and plot total users DoSed on the y-axis in Fig. 6. We perform 10 experiments for each scenario and show averages and standard deviation error bars. To understand the maximum extent of this attack, we use two-hop trust. Even with a small set of such users, the attack is effective for a significant subset of users; however, the attack has diminishing returns as the number of compromised users increases. On average, we see between 3.92% and 10.61% users being finally DoSed for the range of initial users we test. While effective, it is an open question whether such a compromise attack can be successfully mounted at scale.

6 Implementation and Evaluation

To provide further evidence of the feasibility of Moby, we built a prototype client app and evaluated energy costs of running it on a mobile device.

6.1 Implementation Details

We implement the Moby app as a fork of Signal Android app [2] with modifications implementing all functionality of a Moby client. We use Signal's infrastructure as the "secure wide-area communication medium" mentioned in Section 3.1.2. Moby trust establishment is performed with handshake messages sent via Signal. This is performed with all contacts on the first installation of the app and subsequently with any new contacts the user adds. Handshake messages are crafted in such a way that they can be recognized only by other Moby clients and ignored by non-Moby Signal users. Clients respond to handshake messages following the protocol *Moby Handshake* protocol (Section 3.1.3).

The Moby app performs client discovery by advertising Bluetooth MAC addresses via Wi-Fi Direct beacons. Message Exchanges and Post Exchange PSI-CA are done over Bluetooth using MAC addresses obtained in this discovery phase.

These technologies are used in combination to allow the application to discover clients and transmit data without any user interaction. Discovering nearby devices via Bluetooth requires user interaction while setting Wi-Fi Direct beacons does not. Sending messages over Wifi Direct requires user interaction whereas sending them over Bluetooth insecure connections does not. These link layer technologies could be updated in the future; Moby uses them as black-boxes. as they do not affect components of Moby built on top of them.

The asymmetric key cryptography used for a Moby client's public key is RSA with a key size of 2048 bits. The Double Ratchet [40] protocol's Symmetric-key ratchet provides the necessary properties required by the cryptographic material defined in Section 3.1.2. Our implementation uses the Double Ratchet instance used by Signal for a given Moby client, and we ratchet only the Symmetric-key, as break-in resistance (provided by the second ratchet) is not required and out of scope. We use only the encrypted payload and standard MACs that we obtain from this instance of Double Ratchet. We do not add any source or destination identification information; thus, only the client we share this instance of the algorithm with will be able to verify that the message is meant for it. We perform a small study among the authors to verify the functionality of the application and present its open source implementation [9].

6.2 Power Consumption

We now measure the power consumption of our Moby clientimplementation using the Battor [33] power monitor, which provides power readings. We use two Nexus 5 devices, and conducted three measurements on each device (variance among experiment results was sufficiently small to rely on three measurements). We investigated the power consumption of all client operations and found that PSI-CA was the most power-consuming step. We test different values of PSI set sizes and present them in Appendix F; discussing a set size of 100 here. Each test device carries 31464 Joules of energy when fully charged; A PSI-CA operation with 100 input elements consumes 2.5 J of energy and uses 3s of CPU time. This implies 12,586 such exchanges can be performed on a charged phone, assuming no other operations of the phone consume energy. To summarize, we found that mobile devices can handle a large number of message exchanges on a single charge; however, the energy consumption is nontrivial. As such, an implementation needs to carefully consider the input set sizes (to limit energy consumption per exchange). Further, there should be a small number of options (e.g., 25, 50, 100, 200) for input set sizes, to prevent fingerprinting of users by ensuring non-unique set sizes.

Note that measurements were performed on a previous version of Moby that did not include salted fingerprints (before signing) or salt generation. These steps add minor overheads that do not change power consumption of the PSI-CA operation.

7 Discussion

In this section, we discuss some limitations of Moby in order to help clarify what Moby's design achieves.

Binary and Nonbinary Trust: Our system uses binary trust values instead of a trust value that takes on a range of values between zero and one. To justify this decision, we explored the use of a nonbinary trust metric where trust in a user is calculated using communication frequencies. Specifically, we calculate trust in a client as the number of communications with that client divided by the total number of communications, where communications include both calls and text messages. When running simulations with these trust values, we found differences between binary and nonbinary trust that we not statistically significant. Thus we use only binary trust in our simulations.

We analyzed why there were insignificant differences by studying message queues. Trust in Moby is used to decide whether a message should remain in a client's queue or be dropped. For nonbinary trust to affect our simulations, we would need a case where a client's queue is filled entirely with trusted messages. In such cases trust values would affect which messages get to stay and which ones get dropped. However, we observe that such cases are rare; in all cases, the number of untrusted messages always outnumber trusted ones. Thus, trusted messages never compete for space in message queues and nonbinary trust performs the same as binary trust.

Formal Proof of Anonymity: We provide a heuristic, but not formal, proof of anonymity for Moby (Appendix A). This informal proof considers each aspect of anonymity and relies on the non-existence of a global passive adversary. It does not formalize Moby components to prove anonymity properties.

Latency of Moby: Moby aims to provide communication when wide-area networks (*i.e.,* the Internet) are shut down. It does not guarantee message delivery or timely message delivery. Our evaluation of the system tells us that latencies achieved are relatively high (§ 5.5.1). This is an important limitation of the system; however, we believe that some form of secure communication is better than no communication at all.

We also note that higher delays occur for messages that must traverse large geographic distances to reach their destinations. In contrast, there are use cases such as rallies or protests where messages need to be transmitted over short distances. In such scenarios, delays are expected to be small.

Dataset Limitations: As with any trace-driven simulation, there are limitations to our approach. First, we underestimate the set of locations a user visits because we obtain cell tower locations for a user only when they make/receive a call or send/receive a text message. Further, the coarse location granularity based on cell towers means that we cannot precisely identify when two users are within range to share Moby messages; we vary the number of users exchanging messages within an hour at a particular tower to model cases where not all users connected to the same tower can communicate wirelessly during a blackout. Lastly, to construct the "contacts" list for a user, we assume that each user contacted (calls or texts) is part of this list.

Despite these limitations and assumptions, to the best of our knowledge this is largest set of trace data available to us that includes not only information about user mobility but also the set of phone numbers contacted by those users. Even using conservative assumptions about contact lists and the set of users who can forward messages when connected to the same tower, we show that Moby provides reasonably efficient communication during blackout periods.

Movement Patterns During Blackouts: As stated above, we use movement patterns of users collected during periods of Internet connectivity, while evaluating a region under an Internet blackout. Unfortunately, we do not have a way to predict movement patterns of users during blackouts. This is a limitation of our evaluation that we acknowledge but can not address.

Extending Moby to Other Systems: Our approach can be ported to any short-message system that can tolerate delays and dropped messages during blackouts. However, we do not believe that Moby can be extended to arbitrary communication systems. Experience shows that purpose-built anonymity systems (*e.g.,* P2P downloading [25] and voice communication [24]) can outperform general-purpose ones. That said, generalizability remains an important and unsolved goal.

8 Related Work

Anonymous Communication During a Blackout: Closely related works to Moby include Rangzen [28], Briar [11], FireChat [32], and Bridgefy [3], all of which support anonymous communication during blackouts. Rangzen presents a microblogging platform that provides fewer security guarantees than Moby; which in addition to more security properties, provides a messaging platform. Briar provides a messaging system but uses different communication channels to relay these messages, requiring that users be on the same local network for Briar to deliver messages. It also expects users to establish trust manually in person. Moby in contrast, does not require manual trust establishment and provides message delivery even if users are not in proximity to each other. FireChat uses a mobile ad-hoc network to forward messages, but only for a short time window. In addition, FireChat is neither private nor secure [45]. Bridgefy provides an SDK claiming to be private and secure but is broken in practice [7].

Secure and/or Anonymous Networks: Our work is inspired by a wide range of usable, secure and/or anonymous communication tools, such as Tor [18], I2P [1], Signal [2], Herd [24], Dissent [36], and Mesh [4]. Unlike these networks, Moby provides secure, anonymous communication during Internet blackouts.

Secure Encounters: There exist solutions that provide secure ways to discover and recognize devices [27]. Moby does not require its usage as it does not consider cross encounter linkability as a security concern. Further, SDDR is highly user unfriendly as it requires root privileges and breaks usability of applications using it.

Trust Networks: Using a notion of trust among participating nodes has been explored in the past; centralized approaches [13, 43] to do so can be effective, but are not practical in our system due to their centralized nature. Previous decentralized approaches [39, 46, 47] were not designed for communication during blackouts and thus cannot be applied in our network.

Trust-based systems like EigenTrust [22] use global trust values for users while Moby uses local trust. Other reputation-based systems for assigning trust or reputability, *e.g.,* TrustRank [19], are incompatible with Moby because they assume a set of pre-trusted entities.

Anonymity in Moby-Like Networks: Anonymous communications in networks that have similar network architecture like Moby has been studied before [21]. This solution considers a different threat model without the exploration of network attacks of any sort and trusting all participating nodes. They do provide sender-receiver unlinkability but do not guarantee receiver anonymity, which is impractical in malicious environments.

Opportunistic Communication Networks: Moby uses a simple opportunistic routing protocol over a delay-tolerant ad-hoc network to forward messages anonymously during a blackout. While it is possible to use sophisticated routing protocols that optimize performance, we use a relatively straightforward routing protocol. Optimized routing protocols [26, 29] share extra information to make routing more efficient, but this leads to sharing information about users and hence makes these protocols vulnerable to identification attacks. Recent work focused on security, anonymity, and privacy in opportunistic networks [5, 10, 21] consider different threat models in comparison to Moby. Unlike these, Moby is resistant to active network adversaries.

Modeling Mobile Users: Similar to us, prior works use CDR data to model network properties including communication patterns [12], mobility [34, 35], and social ties [23]. We are the first to use such data to evaluate anonymous communications during blackouts.

9 Conclusion

We presented the design, implementation, and evaluation of Moby—a blackout-resistant anonymity network for message communication via mobile devices. Moby's bi-modal design combines a wide-area communication channel and an ad-hoc network to provide secure, attack-resistant communication during disruptions to Internet connectivity. Moby uses a notion of trust built upon the notion that most parties who communicate via calls or text message trust each other more than those who do not. Moby establishes such trust between clients over a secure channel over the Internet during times of connectivity, then uses this trust in Moby's novel ad-hoc network protocol to thwart network adversaries. We implemented and used a custom network simulator and a large set of user mobility and communication traces from a cellular provider to identify the impact of configuration parameters on performance, and demonstrate how trust leads to as much as a 14x improvement in message delivery rates when under a DoS attack. We implemented the Moby client as a proof-of-concept Android app and demonstrated the feasibility of running the app in terms of power consumption.

Acknowledgments

We thank our anonymous reviewers, and our shepherd Saba Eskandarian. This work was supported by the National Science Foundation (Grants: SaTC-1618955 and ProperData SaTC-1955227). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] The invisible internet project (I2P). https://geti2p.net/en/about/intro, 2019.

[2] Signal, 2019. https://signal.org.

[3] bridgefy, 2022. https://bridgefy.me/.

[4] Mesh, 2022. https://mesh.im/.

[5] Paarijaat Aditya, Viktor Erdélyi, Matthew Lentz, Elaine Shi, Bobby Bhattacharjee, and Peter Druschel. Encore: Private, context-based communication for mobile social apps. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '14, pages 135–148, New York, NY, USA, 2014. ACM.

[6] Africa News. Ethiopia restores internet access after shutdown for exams, June 2017. http://www.africanews.com/2017/06/08/ethiopia-restores-internet-access-after-\shutdown-for-exams/.

[7] Martin R Albrecht, Jorge Blasco, Rikke Bjerg Jensen, and Lenka Mareková. Mesh messaging in large-scale protests: Breaking bridgefy. In *Cryptographers' Track at the RSA Conference*, pages 375–398. Springer, 2021.

[8] Anonymized for submission. Private communication with a humanitarian field worker in Syria., September 2017.

[9] Anonymous. Moby proof of concept code, 2019. https://anonymous.4open.science/r/0f48ac77-399c-4ddd-9abe-f28e6782ef4c/.

[10] M. S. Arafath and K. U. R. Khan. Opportunistic sensor networks: A survey on privacy and secure routing. In *2017 2nd International Conference on Anti-Cyber Crimes (ICACC)*, pages 41–46, March 2017.

[11] Biar Project. Briar, 2017. https://briarproject.org/how-it-works.html.

[12] Julián Candia, Marta C González, Pu Wang, Timothy Schoenharl, Greg Madey, and Albert-László Barabási. Uncovering individual and collective human dynamics from mobile phone records. *Journal of physics A: mathematical and theoretical*, 41(22):224015, 2008.

[13] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. Aiding the detection of fake accounts in large scale social online services. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 15–15. USENIX Association, 2012.

[14] H. Corrigan-Gibbs, D. Boneh, and D. Mazières. Riposte: An anonymous messaging system handling millions of users. In *2015 IEEE Symposium on Security and Privacy*, pages 321–338, May 2015.

[15] Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Fast and private computation of cardinality of set intersection and union. In *International Conference on Cryptology and Network Security*, pages 218–231. Springer, 2012.

[16] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In *International Conference on Financial Cryptography and Data Security*, pages 143–159. Springer, 2010.

[17] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In *International Workshop on Privacy Enhancing Technologies*, pages 54–68. Springer, 2002.

[18] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[19] Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with trustrank. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 576–587. VLDB Endowment, 2004.

[20] Human Rights Watch. India: 20 internet shutdowns in 2017, June 2017. https://www.hrw.org/news/2017/06/15/india-20-internet-shutdowns-2017.

[21] Rob Jansen and Robert Beverly. Toward Delay Tolerant Network Anonymity: Threshold Pivot Scheme. In *Proceedings of the Military Communications Conference (MILCOM 2010)*, 2010.

[22] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 640–651, New York, NY, USA, 2003. ACM.

[23] Márton Karsai, Nicola Perra, and Alessandro Vespignani. Time varying networks and the weakness of strong ties. *Scientific Reports*, 4(1), May 2014.

[24] Stevens Le Blond, David Choffnes, William Caldwell, Peter Druschel, and Nicholas Merritt. Herd: A Scalable, Traffic Analysis Resistant Anonymity Network for VoIP Systems. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 639–652, New York, NY, USA, 2015. ACM.

[25] Stevens Le Blond, David Choffnes, Wenxuan Zhou, Peter Druschel, Hitesh Ballani, and Paul Francis. Towards efficient traffic-analysis resistant anonymity networks. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, page 303–314, New York, NY, USA, 2013. Association for Computing Machinery.

[26] Jeremie Leguay, Timur Friedman, and Vania Conan. Evaluating mobility pattern space routing for DTNs. *arXiv preprint cs/0511102*, 2005.

[27] Matthew Lentz, Viktor Erdélyi, Paarijaat Aditya, Elaine Shi, Peter Druschel, and Bobby Bhattacharjee. SDDR: light-weight, secure mobile encounters. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 925–940, 2014.

[28] Adam Lerner, Giulia Fanti, Yahel Ben-David, Jesus Garcia, Paul Schmitt, and Barath Raghavan. Rangzen: Anonymously Getting the Word Out in a Blackout. *arXiv:1612.03371 [cs]*, December 2016. arXiv: 1612.03371.

[29] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic Routing in Intermittently Connected Networks. In *Service Assurance with Partial and Intermittent Resources*, Lecture Notes in Computer Science, pages 239–254. Springer, Berlin, Heidelberg, 2004. DOI: 10.1007/978-3-540-27767-5_24.

[30] New Scientist. Earthquake shakes the internet, January 2007. https://www.newscientist.com/article/mg19325852-300-earthquake-shakes-the-internet/.

[31] New York Times. After a Cyberattack, Germany Fears Election Disruption, December 2016. https://www.nytimes.com/2016/12/08/world/europe/germany-russia-hacking.html?mcubz=3.

[32] Open Garden. Firechat, 2018. https://www.opengarden.com/firechat/.

[33] Aaron Schulman, Thomas Schmid, Prabal Dutta, and Neil Spring. Phone power monitoring with battor. In *17th ACM International Conference on Mobile Computing and Networking (MobiCom 2011)*, 2011.

[34] M Shahzamal, M F Parvez, M A U Zaman, and M D Hossain. Mobility Models for Delay Tolerant Network: A Survey. *International Journal of Wireless & Mobile Networks*, 6(4):121–134, August 2014.

[35] C. Song, Z. Qu, N. Blumm, and A.-L. Barabasi. Limits of Predictability in Human Mobility. *Science*, 327(5968):1018–1021, February 2010.

[36] Ewa Syta, Henry Corrigan-Gibbs, Shu-Chun Weng, David Wolinsky, Bryan Ford, and Aaron Johnson. Security analysis of accountable anonymity in dissent. *ACM Trans. Inf. Syst. Secur.*, 17(1), August 2014.

[37] Telegraph. Unprecedented cyber attack takes Liberia's entire internet down, November 2016. http://www.telegraph.co.uk/technology/2016/11/04/unprecedented-cyber-attack-takes-liberias-\entire-internet-down/.

[38] The Guardian. Iraq shuts down the internet to stop pupils cheating in exams, May 2016. https://www.theguardian.com/technology/2016/may/18/iraq-shuts-down-internet-to-stop-pupils-\cheating-in-exams.

[39] N. Tran, J. Li, L. Subramanian, and S. S. M. Chow. Optimal Sybil-resilient node admission control. In *2011 Proceedings IEEE INFOCOM*, pages 3218–3226, April 2011.

[40] Trevor Perrin and Moxie Marlinspike. The Double Ratchet Algorithm, November 2016.

[41] Nirvan Tyagi, Yossi Gilad, Derek Leung, Matei Zaharia, and Nickolai Zeldovich. Stadium: A distributed metadata-private messaging system. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 423–440, New York, NY, USA, 2017. ACM.

[42] Amin Vahdat, David Becker, et al. Epidemic routing for partially connected ad hoc networks. 2000.

[43] W. Wei, F. Xu, C. C. Tan, and Q. Li. SybilDefender: A Defense Mechanism for Sybil Attacks in Large Social Networks. *IEEE Transactions on Parallel and Distributed Systems*, 24(12):2492–2502, December 2013.

[44] Alma Whitten and J Doug Tygar. Why johnny can't encrypt: A usability evaluation of pgp 5.0. In *Proceedings of the 8th USENIX Security Symposium*, volume 348, Washington, D.C., 1999.

[45] WIRED. Protesters adore firechat but it's still not secure, 2014. http://www.wired.co.uk/article/firechat-app-hong-kong-protesters.

[46] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. SybilLimit: A Near-Optimal Social Network Defense Against Sybil Attacks. *IEEE/ACM Transactions on Networking*, 18(3):885–898, June 2010.

[47] Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham D. Flaxman. SybilGuard: Defending Against Sybil Attacks via Social Networks. *IEEE/ACM Trans. Netw.*, 16(3):576–589, June 2008.

A Heuristic Proof of Anonymity

Moby provides end-to-end encryption, forward secrecy and sender-receiver anonymity. To provide these features, we begin with a protocol that provides end-to-end encryption and forward secrecy. In the case of our implementation, we use the Double Ratchet (DR) algorithm. We make the following modifications to provide sender-receiver anonymity.

Sender Anonymity: The Moby network is intended to be deployed as a geographically distributed ad-hoc wireless network, making a global adversary that can observe the entire network impractical. Thus, in a network where all senders cannot be observed, a participant that introduces a message to the network cannot be distinguished from one that is forwarding it. If a network observer sees a client that sends out a message, it cannot tell if this client is the sender or a forwarder. Therefore, messages are sender anonymous as an adversary cannot identify who the sender of a given message is.

An attacker could potentially observe TTLs and try to use them to infer senders. We add noise to the TTLs in messages to prevent leaking information about senders.

Receiver Anonymity: Prior to a blackout, Moby clients establish trust with each other, and as part of trust establishment they receive cryptographic material to be used for blackout communications. On performing encryption with this material, a client produces an *encrypted payload* and an associated *MAC* for the payload. Without knowledge of the key shared between clients, an adversary cannot decrypt this payload, nor can it compute the *MAC* associated with it. Thus, Moby messages lack information pointing to who the receiver of a payload is, and provide receiver anonymity.

To further ensure receiver anonymity, receivers of messages behave in a similar manner to clients that don't receive messages, by holding received messages in their message queues to prevent observers from detecting message reception.

Extensibility: Moby defines its cryptographic components in a way that developers can easily modify them without impacting other aspects of the Moby protocol. Drop in replacements for DR, with certain modifications, could be used while still following the protocol defined by Moby to attain the guarantees it provides in terms of network communications.

```
1: procedure ProcessMessageQueue($newMessages$)
2:   for $message$ in $newMessages$ do
3:     for $contact$ in $allContacts$ do
4:       $MACKey \leftarrow MAC\ key\ for\ contact$
5:       $computedMAC \leftarrow HMAC(messagePayload, MACKey)$
6:       if $computedMAC = messageMAC$ then
7:         $New\ message\ received$
8:       else
9:         $Add\ message\ to\ local\ queue\ after\ update$
10:      end if
11:    end for
12:  end for
13: end procedure
```

Fig. 7. Steps involved in checking if a message received via a message exchange is meant for a Moby client.

B Receiving a Moby Network Message

When a client receives a message via the Moby network, it needs to check if it is the intended destination of that message. Most messaging systems use destination identifiers to accomplish this, but Moby messages lack such information to provide sender anonymity. Thus, clients follow the algorithm presented in Figure 7. For every message in the set of new messages, the receiving client checks every session it shares with its contacts and computes *HMAC*s using the message payload and each *MAC Key*. If the computed *HMAC* matches the one attached to the message, the client knows that the message was meant for it, and who the sender is, based on the *MAC Key* that resulted in the match.

Thus, without attaching identifiers to a Moby message, Moby clients can figure out who senders of the message are, and identify that they were the intended destination. Although this process is computationally expensive, it is worth using to attain sender/receiver anonymity.

C Moby Simulation Parameters

Table 2 summarizes the parameters we use in Moby simulations.

D Trade-Offs Between Delivery Ratios and Latencies

Most messaging systems strive for high delivery ratios and low latency. While the previous paragraphs considered delivery ratios and latencies in isolation, we now investigate to what extent we can tune Moby to offer both reasonably high delivery ratios and low latency.

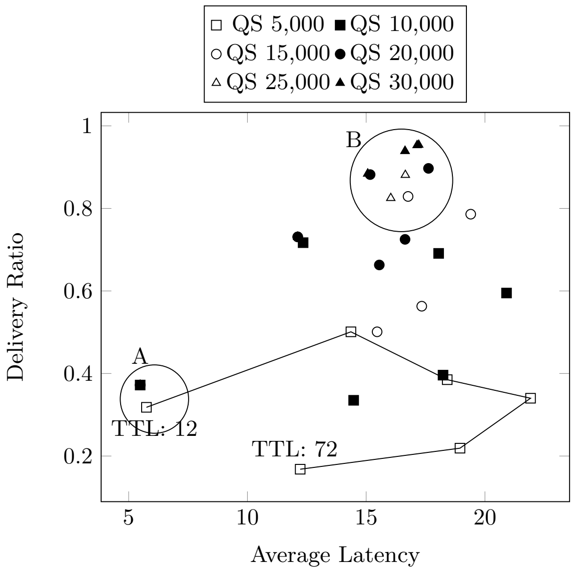

Fig. 8. Scatter plot of average latency versus delivery ratio for various simulation parameters. Cluster A indicates configurations that have the lowest latency (due to low TTLs) and Cluster B identifies those with reasonably good trade-offs between latency and delivery ratio.

To analyze the effects of varying user queue size and time to live (TTL) on both delivery ratio and average message latencies, we plot delivery ratios on the y-axis and average message latencies on the x-axis in Fig. 8. We present metrics for simulations with a liveness of 4, all other liveness values resulted in similar trends. Note that values closer to the top left of the figures are generally considered better: those result in both higher deliver ratios and lower latency.

To help with reading the figure, we manually annotated the graph with two clusters and a curve. Starting with Cluster A, we can clearly identify cases where there is the lowest latency—but this comes at the cost of low delivery ratios. Such cases come from simulations using a TTL of 12, for all queue sizes, 5,000 to 30,000, and the low latency is easily explained by the low TTL.

Considering Cluster B, we find cases that have reasonably high delivery ratios (>0.8) with latencies near the middle of the range (15-17.5 hours). These high delivery ratios are observed for high queue size simulations: 15,000 (TTL: 36), 20,000 (TTL: 36, 48), 25,000 (TTL: 36 - 72), and 30,000 (TTL: 36 - 72). The common factors for such performance are relatively large queue sizes (to increase delivery ratios) with moderately sized

Parameter Name	Description	Values	Reasoning
Number of Days	The total number of days the simulation runs	3 days	Captures the functionality of the network
Number of Messages	The total number of messages sent over the duration of the simulation	30,000	Based on number of messages sent by users
Cooldown	The duration of the simulation (towards the end) when messages are not sent out	24 hours	Longer cooldowns lead to redundant results
Liveness	The minimum number of hours a user needs to participate in the dataset to be considered in the simulation	1, 2, 4, 6, 8, 12	Larger values lead to smaller sets of eligible users
Queue Size	The size of a message queue for a simulated user	5000, 10000, 15000, 20000, 25000, 30000	Larger queue sizes are capped by the number of messages sent
Time to Live	The amount of time a message is alive in the simulation	12, 24, 36, 48, 60, 72	Capped by the simulation duration
Number of DoS Messages	The number of messages sent by a malicious entity in an hour per cell tower	0, 2, 4, 6, 8, 10	Higher values lead to redundant results
Contacted Threshold	For a user, the minimum number of times another user needs to be contacted to be considered part of its contact list	1, 2, 3, 4	Higher values lead to redundant results
Hop Count	The maximum number of hops away a user can be to be considered trusted	0, 1, 2	Higher values means trusting users who are in general complete strangers

Table 2. Simulation parameters and corresponding values.

TTLs (36 hours or higher for larger queue size) to tame queue buildup.

We next highlight an interesting trend using the line linking hollow squares in Fig. 8. This represents TTL 12–72 (increments of 12) for the queue size 5,000. We notice that keeping a queue size constant, increasing the TTL first increases delivery ratios and then leads to a drop in Fig. 3; similarly, varying parameters this way leads to an increase in average latency and then a drop in the average latency. The drop in delivery ratio also results in a drop in average latency as messages get delivered earlier in the simulation at which point queues get filled with messages that never get delivered.

E Effect of Exchange Probability on Delivery Ratios

One of the limitations of our dataset is that we lack precise geolocation information for users. We, however, do possess user to tower mappings for when these users send/receive text messages or call/receive calls. Since we are unable to model fine-grained proximity between users at the same tower and use this to estimate the probability of successful messages exchanges between users, we instead use a probabilistic approach. Namely, we uniformly randomly sample a fraction of pairs of users connected to a tower during an hour, and simulate the case where only those pairs of users exchange messages during that hour. We then explore the effects of varying this random fraction of pairs exchanging messages on delivery ratios for epidemic routing. We explore the following percentages: 1, 2, 4, 8, 10, 50, and 100%.

	Exchange Probability						
TTL	100	50	10	8	4	2	1
12	.7318	.7282	.6193	.5807	.3862	.1281	.134
24	.9421	.9402	.8959	.8762	.7438	.3687	.366
36	.9688	.9678	.9401	.9275	.8417	.5561	.957
48	.9822	.9817	.9655	.9572	.8940	.6521	.1469
60	.9842	.9837	.9693	.9618	.9071	.6881	.195
72	.9847	.9842	.9709	.9638	.9111	.6961	.2062

Table 3. Drop in delivery ratios as Exchange Probability is reduced.

Table 3 shows the drop in delivery ratios as exchange probability is reduced, for various TTLs. We observe a negligible drop in performance from 100% to 50% (under 0.5%) with larger drop as the percentage is decreased; we see critical drops in performace for exchange probabilities under 5%. We observe that higher TTLs have more resistance to low exchange probabilities, notice that even 8% yields high delivery ratios for TTLs 36 and higher. These trends are understandable as fewer message exchanges would naturally lead to poorer message propagation, which in turn reduces delivery ratios. Whereas higher TTLs allow messages to stay in circulation longer, that counteracts low exchange probabilities. Interestingly, even at low exchange probabilities (*e.g.,* 10%), we see that the performance of the network drops by under 5% for TTL 24 and even lower (under 2%) for higher TTLs. This supports the viability of a large-scale deployment of Epidemic routing, and thus Moby.

F PSI-CA Power Consumption

We measure power consumption of our Moby client implementation on two Nexus 5 devices using the Battor power measurement device. We find that PSI-CA is the most CPU and power intensive step of the client and thus measure consumption for different intersection set sizes. We present results for this in Fig. 9, with PSI set sizes on the x-axis and the CPU time (left) and energy consumed (right) are the y-axes. We observe a clear trend indicating substantially more CPU and power consumption for larger set sizes, and the consumption scales approximately linearly with set size. Thus, a way to limit the power consumption of Moby is to either eliminate PSI-CA entirely, or use reasonably small set sizes (*e.g.,* pick a subset of 100 contacts as input).

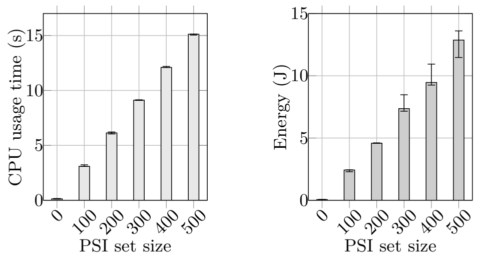

Fig. 9. Evaluation of the CPU/power consumed by the prototype performing a PSI operation for various PSI set sizes.