Received April 17, 2020, accepted May 1, 2020, date of publication May 7, 2020, date of current version May 21, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.2993062*

# Secure and Anonymous Communications Over Delay Tolerant Networks

**SPIRIDON BAKIRAS[1], (Member, IEEE), ERALD TROJA[2], XIAOHUA XU[3], AND JULIANO FISCHER NAVES[4]**

[1]Division of Information and Computing Technology, College of Science and Engineering, Hamad Bin Khalifa University, Doha 34110, Qatar
[2]Department of Computer Science, School of Professional Studies, St. John's University, Queens, NY 11439, USA
[3]Department of Computer Science, College of Computing and Software Engineering, Kennesaw State University, Marietta, GA 30060, USA
[4]Federal Institute of Rondônia, Vilhena 76980, Brazil

Corresponding author: Spiridon Bakiras (sbakiras@hbku.edu.qa)

**ABSTRACT** Today's society has a fundamental need for security and anonymity. Well suited, real-life scenarios such as whistleblower reports, intelligence service operations, and the ability to communicate within oppressive governments, call for such fundamental needs. The contribution and focus of this paper is the study of anonymous communications in the context of Delay Tolerant Networks (DTNs). Current literature achieves anonymity via mechanisms that are built around the onion routing paradigm which, unfortunately, is vulnerable to malicious nodes. Instead, our work introduces a novel message forwarding algorithm that delivers messages, from source to destination, via a random walk process. As such, our protocol does not list the intermediate nodes along the route's path and, therefore, enhances significantly the anonymity of the underlying communications. We propose two different approaches for encrypting the exchanged messages. The first one is based solely on public key cryptosystems and is, thus, suitable for short, SMS-style messaging. The second one is a hybrid solution that combines both public and symmetric key cryptography and is targeted towards large multimedia messages, such as images or video. Through extensive simulation experiments, we show that our proposed anonymous routing protocol achieves high message delivery rates, while using modest computational resources on the mobile devices.

**INDEX TERMS** Anonymous communications, confidentiality, delay tolerant networks, random walks.

## I. INTRODUCTION

The advancement of cybersecurity education and the ability to leverage high-tech vulnerability assessment, penetration testing, and software exploitation tools, have turned privacy breaches into commonplace events. Major data breaches such as the Facebook [1] and Equifax [2] incidents, demonstrate that the storage of unencrypted data on servers could have catastrophic results. In addition, NSA's widely used techniques of electronic communication surveillance [3], illustrate that plaintext data in transit could lead to personal information disclosure. Finally, oppressive governments may not only be interested in the contents of a private communications, but also in the identities of the communicating parties, which could potentially identify citizens that share similar anti-government sentiments. Consequently, we as a society, have reached a point where secure and anonymous communications have become critical. To this end, *Tor* (the onion router) [4] is the de facto anonymization platform for online communications. It is mainly targeted towards anonymous web browsing, and enjoys a very diverse user population that ranges from the military to media, activists, and even family and friends.

The technology behind Tor originates from Chaum's research on mix nets [5]. In particular, Tor leverages the concept of layered encryption (onion routing), where each encryption layer corresponds to one of the pre-determined routing nodes along the path from the source to the destination. As the message travels through its route, every intermediate node removes one of the encryption layers and, in doing so, discloses the next hop towards the destination. Figure 1 shows an example of onion routing, where Alice wants to send a message to Bob anonymously. She first selects three random Tor servers (from a public list) and then encrypts her message recursively, using the public keys of the selected nodes. At each node, Alice's message is mixed with the rest of

The associate editor coordinating the review of this manuscript and approving it for publication was Chih-Min Yu.

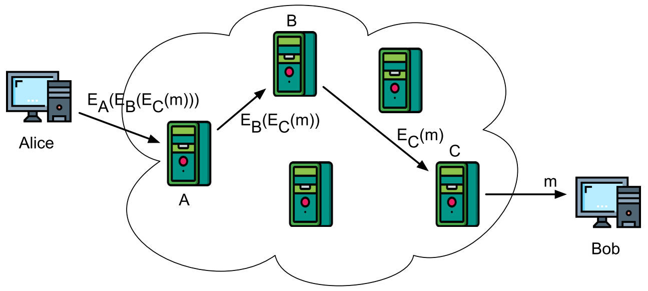


**FIGURE 1.** **Onion routing.**

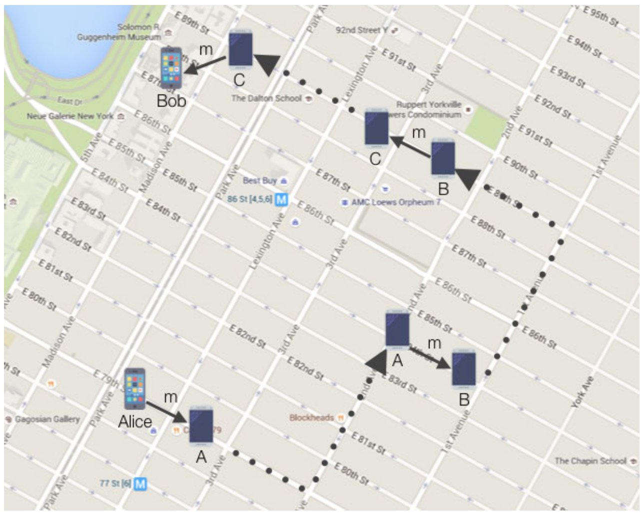


**FIGURE 2.** **DTN routing.**

the Tor traffic, in order to hide the next destination from the adversary. This type of cryptographic-based source routing mechanism is employed by the majority of the currently operational anonymous networks. The main reason behind this wide-scale adoption is the efficiency and low end-to-end delay of onion routing, which is essential in web browsing applications. However, onion routing is known to be very vulnerable to malicious nodes, since every compromised node can disclose one edge of the route between the source and the destination. In the worst case scenario, when all intermediate nodes are compromised (e.g., nodes $A$, $B$, and $C$ in Figure 1), the underlying conversation ceases to be anonymous.

Delay Tolerant Networks (DTNs) [6] provide an alternative architecture for implementing anonymous communications. DTNs employ an opportunistic routing paradigm, where messages are randomly bounced among participating nodes, in a store-carry-and-forward fashion. Therefore, in light of their peer-to-peer (P2P), infrastructure-less nature, DTNs are more resilient against adversarial nodes. Indeed, Tor is vulnerable to targeted attacks against its infrastructure. An adversary may compromise a number of Tor routers (thus breaking anonymity) or launch DDoS attacks to take the entire Tor network offline. On the other hand, DTNs operate without the involvement of a fixed communication infrastructure and are, therefore, extremely difficult to shut down or compromise. DTN-based messaging applications have gained a lot of attention recently, and have been used to prevent the monitoring or blocking of communications by state actors [7], [8]. Figure 2 illustrates the concept of DTN routing. Unlike typical messaging applications that utilize a dedicated server, DTN routing relies on a best-effort approach to deliver the message from Alice to Bob. First, Alice comes in contact with node $A$ and forwards Bob's message to it. Similarly, the message is subsequently forwarded to nodes $B$ and $C$, before it finally reaches Bob when he moves in close proximity to node $C$.

Unfortunately, previous research on anonymous communications in DTNs [9]–[11] also leverages the onion routing paradigm and, specifically, a method called *group* onion routing. More concretely, all nodes are split into groups, and the source node selects a set of groups that the message has to traverse before reaching the destination. Furthermore, these schemes employ a *trusted* key generator that generates the (individual or per group) private keys that are necessary to establish the group onion routes. Obviously, this is a major security risk that implies a tremendous amount of trust on the security of the third-party key generator. To this end, our work abandons the standard onion routing paradigm and, instead, proposes a novel distributed approach to anonymous communications. In particular, our methods build upon the opportunistic communication of DTNs to deliver messages via the conventional store-carry-and-forward process. Compared to onion routing, our protocols provide more stringent anonymity guarantees, since the end-to-end route of a message is entirely random and malicious nodes cannot infer any information about the next hop of a given message.

In our earlier work [12], we designed a protocol that relies exclusively on public key cryptography. As such, it is only suitable for the exchange of short messages that can fit into a few ciphertexts, because every message has to be re-randomized at each forwarding step. (Note that, re-randomization is expensive for public key ciphertexts.) In this paper, we extend our basic protocol and introduce a hybrid solution where (i) the message is encrypted with a symmetric cipher and (ii) the symmetric keys used for message re-randomization are communicated via public key ciphertexts. We conducted a thorough experimental evaluation of our methods using real-life datasets, and our results indicate that the proposed forwarding algorithm attains high message delivery rates, while using only modest computational resources at the mobile devices.

The remainder of the paper is organized as follows. Section II gives an overview of previous research on anonymous communications and Section III presents the details of our anonymous messaging system that is based on public key cryptography. Section IV introduces the hybrid protocol that leverages symmetric cryptography and Section V discusses the anonymity properties of our design. Section VI presents the results of our experimental evaluation and Section VII concludes our work.

## II. RELATED WORK

There is an abundance of research work on onion routing networks, including Ref. [13]–[18]. Their differences lie on the specific cryptographic constructions of the underlying encryption layers. In addition to theoretical research, various anonymization networks (besides Tor) have been implemented into actual systems, including Mixmaster [19] and Mixminion [20] for anonymous email, and I2P (Invisible Internet Project) [21] for anonymous messaging, web browsing, blogging, email, and file sharing. However, all the above systems are deployed on networks with a dedicated infrastructure.

In the wireless domain, anonymity has been addressed in the context of routing protocols for mobile ad-hoc networks (MANETs). For example, MASK [22] introduces an anonymous neighborhood authentication protocol based on bilinear maps, which is then used to establish anonymous, on-demand routes between communicating nodes within the MANET. ANODR [23] is an anonymous on-demand routing protocol that provides route anonymity for the source and destination nodes, and also protects the location privacy of the transmitting nodes. Wu and Bertino [24] proposed the zone-based anonymous positioning (ZAP) routing protocol, where nodes are assigned into anonymity zones that hide the identities of the communicating nodes with the use of local flooding. In general, the main limitation of all anonymous routing protocols is the underlying assumption that there is a communication path between any source-destination pair. Furthermore, routing protocols incur a significant amount of overhead to maintain and update the network graph information, especially when nodes are highly mobile. Instead, in our work, we do not make any assumptions regarding network connectivity, which is the main motivation behind delay tolerant networks.

The first protocol that addresses security and anonymity in DTNs was designed by Kate *et al.* [25], as part of a study involving Internet access in remote areas. Their approach leverages the concept of identity-based cryptography (IBC) [26] to implement an anonymous authentication protocol, which gives users the ability to authenticate via aliases instead of their real identities. The drawback of this method is that it requires a *trusted* party, namely a key generator, that generates and distributes all private keys to the end users. The centralized key generator is an easy target for attackers and, if compromised, it could lead to the disclosure of all private keys, thus breaking the security and anonymity of every user in system.

Jansen and Beverly [9] introduce the threshold pivot scheme (TPS), which modifies the onion routing protocol in order to make it applicable to the infrastructure-less environment of DTNs. The paper suggests a *group* onion routing algorithm that uses a threshold secret sharing scheme [27] towards the distribution the encryption key among the different groups. Therefore, the only way to learn a message's destination is by reconstructing the secret through nodes in at least $\tau$ distinct groups. TPS has some obvious disadvantages. First, a number of colluding or malicious nodes can reveal all the receivers' identities, once their combined memberships span at least $\tau$ different groups. On the other hand, if their memberships span across all groups, they can also identify the message's sender by following its path through the various groups. Second, the protocol requires every group to maintain a unique public/private key pair, which is very hard to accomplish without a trusted third party.

ARDEN [10] is similar to TPS, in that it employs group onion routing to provide anonymization of messages. However, in ARDEN, the groups are defined dynamically. In particular, the sender node utilizes attribute-based encryption (ABE) [28] to build random groups that are determined by the IDs of the underlying DTN nodes. As such, the probability that an adversary has complete control over a randomly generated group is relatively low. Nevertheless, ARDEN still has some severe drawbacks. First, ABE requires a trusted entity to generate and distribute the secret keys to all the network nodes. As previously mentioned, the centralized key generator is a single point of failure, whose compromise could give an adversary full decryption privileges for all the messages that are routed inside the DTN. Furthermore, the sender node must have knowledge of a significant number of user IDs that are currently active in the system, in order to construct the group onion route. This task implies an intricate network maintenance mechanism that may impose a non-trivial overhead in the system. Lastly, ABE constructions are computationally expensive, and as such, they may incur a considerable burden on the mobile devices.

Sakai *et al.* [29] analyze the security of onion-based anonymous routing protocols for DTNs. They introduce a metric called path anonymity and apply it on anonymous routing protocols for both single-copy and multi-copy message forwarding. The same authors later propose a framework of anonymous routing (FAR) [11] that combines the concepts of group onion routing and epidemic or zone-based routing. Specifically, the source node first establishes a set of group onion routers, and then messages are forwarded inside the network with an anonymous restricted epidemic routing protocol. Finally, Lu *et al.* [30] introduce an anti-localization routing protocol (ALAR) for anonymous message delivery in DTNs. The idea is to divide each message into a number of segments, and then send each segment to many different receivers. However, the objective of their work is not to hide the identities of the communicating nodes, but rather to protect the location privacy of the sender.

Besides anonymity, there is some recent research work that addresses the privacy concerns of DTN routing. For example, PRIVO [31] models a DTN as a time-varying neighboring graph where edges correspond to the neighboring relationship among pairs of nodes. It ensures privacy by protecting each node's sensitive information, even if it has to be processed elsewhere. One of the critical ingredients in their approach is the amount of information that each node is willing to share, i.e., there is a trade off between the amount of shared information and the performance of the routing protocol. ePRIVO [32] proposes an enhanced and modified version of PRIVO, targeted towards the Vehicular Delay Tolerant Network (VDTN) domain. Its goal is to enable vehicles to take routing decisions while keeping their information private. Finally, Jiang *et al.* [33] propose a privacy-preserving protocol for utility-based routing (PPUR) in DTNs. In their approach, DTN nodes utilize routing utility values based on several metrics, such as encounter time, frequency, etc. However, the encounter records are collected via pseudo-IDs and are forwarded to a trusted authority. Messages are then forwarded among nodes based on the real utility values that are computed by the trusted authority.

## III. ANONYMOUS MESSAGING SYSTEM

In the following sections we introduce the specifics of our message anonymization system. We first present the underlying threat model, followed by a detailed description of the various system ingredients, including key management, message forwarding, and cryptographic primitives.

### A. THREAT MODEL

We designed our system to thwart attacks from both *passive* and *active* adversaries. On one hand, passive adversaries are allowed to eavesdrop on all communications within the DTN environment. Our assumption is that the adversaries are bound to polynomial running times and, as such, they cannot break the public/symmetric encryption schemes that are applied on individual messages. On the other hand, active adversaries are allowed to compromise a fraction of the honest nodes and force them into malicious behavior. These malicious nodes can launch a number of attacks against legitimate users, by forwarding to them specifically crafted messages, including replay messages, and observing their subsequent communication patterns. The goal of such attacks is to identify any pair of users that are engaged in covert communications. Here, we are not interested in defending against denial-of-service (DoS) attacks, since they are not of a cryptographic nature, and thus, outside the scope of this work.

### B. KEY MANAGEMENT

The key operation of an anonymous router is to make the incoming and outgoing messages indistinguishable, in order to hide the end-to-end path. In onion routing, this is accomplished through the use of layered encryption, i.e., once a router removes one encryption layer, the message can not be linked to its previous instance. In this work, we employ a single encryption layer and provide indistinguishability via message re-randomization. To achieve this goal, we leverage end-to-end encryption that is implemented through the existing public key infrastructure (PKI) [34]. More concretely, we make the valid assumption that the sender node is in possession of the public key of the recipient, which is then used to encrypt the transmitted messages. As such, no other entity (honest or malicious) is able to decrypt the messages and compromise the communication.

For this approach to work in practice, we need a secure method for users to obtain the public keys of the recipient nodes. A possible solution is to store all keys in a public database and allow users to download the keys on demand. Nevertheless, to preserve the anonymity of the recipient node, the sender has to either download the entire database or query the database in a private manner, e.g., using private information retrieval (PIR) protocols [35]. Moreover, all uploaded keys must be signed by a trusted certification authority (CA), so that users can verify their authenticity and thwart man-in-the-middle attacks. It should be noted that, unlike existing work, all users are responsible for generating their own public/private key pairs without the involvement of a trusted third party. Another possibility is that users opt to exchange their public keys offline, i.e., through a secure two-party protocol. This solution assumes that users are familiar with secure key-exchange protocols, such as Diffie-Hellman.

### C. MESSAGE FORWARDING

At the core of our anonymization network lies a novel routing protocol that randomly bounces messages among the roaming nodes until they are delivered to their destination (random walk). Nevertheless, to protect anonymity against an adversary with a global view of all exchanged messages, several actions are necessary. First, all messages should be anonymized, i.e., the identity of the recipient should not appear on the message itself. Second, every intermediate node should obfuscate all outgoing messages (via public key re-randomization techniques), so that they are indistinguishable to an adversary. Finally, all messages should have an identical size, i.e., smaller messages should be padded accordingly, while larger messages should be fragmented and delivered via multiple chunks.

Suppose that Alice wants to send a message $m$ to Bob anonymously. She initially creates a packet[1] $P$ that contains the following fields ('|' denotes concatenation):

$$P = \langle nym, \mathcal{PK}, m, H(nym|\mathcal{PK}|m)\rangle$$

$\mathcal{PK}$ is a *fresh* public key is not part of the public key database, i.e., it is not associated with Alice. The reason for including $\mathcal{PK}$ is to hide Alice's real identity from Bob. As such, if Bob decides to send a reply or acknowledgment message back to Alice, he will use this fresh public key to encrypt the message. Similarly, Alice associates a pseudonym ($nym$) to public key $\mathcal{PK}$, instead of her real name.[2] Moreover, to identify and avoid attacks that can modify messages, Alice will compute and attach the message digest $H(\cdot)$ of the given data packet. This message digest can also be leveraged by Bob in order detect duplicate (or replayed) messages. $P$ is finally encrypted with Bob's public key, prior to being communicated out onto the network.

[1]Or *bundle* in the DTN terminology.

[2]When two users know each other but want to hide their communication from an adversary, they may use their real identities and public keys instead.

DTNs utilize a store-carry-and-forward routing protocol. In particular, mobile nodes store messages in a local buffer $B$ while roaming around the network. Once a node comes within communication range with another node that it has not encountered before, the two will swap (part of) their stored messages and then move on with their intended trajectories. This forwarding process stops when a node receives a message that is addressed to itself and is, thus, removed from its local buffer. The most straightforward message forwarding algorithm is flooding, also known as *epidemic* routing. Here, mobile nodes exchange replicas of their entire buffers with every other node they run into. Clearly, epidemic routing incurs very high storage and processing costs, which has led to the development of more efficient alternatives, including spray routing [36], PRoPHET [37], RAPID [38], and many others.

In our problem setting, current DTN routing algorithms are confronted with the following challenges: (i) messages do not contain the destination address or any time-to-live (TTL) information, (ii) it is impossible for a node to identify and remove duplicate messages, and (iii) every forwarded message incurs a significant computational overhead, due to the public key cryptographic operations that are involved. Challenges (ii) and (iii) call for the use of *multi-copy* routing algorithms, such as Spray and Wait [36], which create a fixed number $k$ of copies for each generated message. This is essential because, without a mechanism that identifies duplicate messages, a flooding style algorithm (such as epidemic routing) will create an exponential number of copies that have to go through expensive re-randomization operations at each hop. Challenge (i) also prohibits direct transmission routing [39] (i.e., direct delivery of a message to its recipient), as well as smart algorithms that leverage node history/location in the forwarding algorithm [37].

In order to address those issues, we incorporate the following features in our design. First, when a node wishes to send a new message, it constructs $k$ copies and places them randomly into its outgoing buffer $B$. Furthermore, to control the overall overhead, the buffer length $|B|$ is fixed to a relatively small size, and new incoming packets are allowed only if there is sufficient buffer space. Lastly, in order to reduce the computational cost of ciphertext re-randomization (and also increase the adversary's uncertainty with regards to the routing process), only a fraction $f$ of the outgoing buffer $B$ is swapped between two communicating nodes. Moreover, the outgoing messages are immediately removed at the sender. Algorithm 1 describes the message forwarding logic of our system. Observe that, when a node receives a batch of messages from another node, it has to inspect all of them in order to determine whether they are destined to itself (lines 3–5). This is necessary, because messages do not include destination addresses. Ownership is verified by decrypting a ciphertext with a known plaintext value (as discussed in the following section) and confirming that the decrypted value is indeed correct.

**Algorithm 1** Message Forwarding Algorithm

```
1:  procedure Receive-Buffer($Q$)
2:      // Input: A buffer $Q$ received from a connected peer
3:      for each packet $P$ in $Q$ do
4:          if $decrypt(P)$ = true then
5:              store $P$ for further processing;
6:          else
7:              $B.enqueue(P)$;
8:          end if
9:      end for
10: end procedure
11:
12: procedure Send-Buffer($B$)
13:     // Input: Local buffer $B$
14:     Initialize an empty buffer $Q$;
15:     for each packet $P$ in $B$ do
16:         $u \overset{R}{\leftarrow} [0, 1)$;
17:         if $u < f$ then
18:             $Q.enqueue(P)$;
19:             $B.remove(P)$;
20:         end if
21:     end for
22:     $Q.randomize()$;
23:     Send $Q$ to the connected peer;
24: end procedure
```

### D. CHOOSING A SUITABLE CRYPTOSYSTEM

Our message forwarding protocol necessitates a public key cryptosystem that allows for ciphertext re-randomization without knowledge of the underlying public key. In this work, we leverage the ElGamal cryptosystem [40] that has this desirable property. The operation of the cryptosystem is summarized as follows.

1) **Initialization**: Let $p = 2q + 1$ be a safe prime, and $\mathbb{G}$ be a cyclic group of prime order $q$ under multiplication modulo $p$. Let $g$ be a generator of $\mathbb{G}$. All users in the system share the same public parameters $(\mathbb{G}, g, q, p)$.
2) **Key generation**: Choose a private key $x$ uniformly at random from $\mathbb{Z}_q$, and set the public key $h = g^x$.
3) **Encryption**: Given a message $m \in \mathbb{Z}_q^*$, choose a uniformly random $r \in \mathbb{Z}_q$ and compute the ciphertext $(c_1, c_2) = (g^r, m \cdot h^r)$. Note that all ciphertexts belong to the same group, regardless of the underlying public/private key pair.
4) **Decryption**: Compute $m = c_2/c_1^x$.

To allow intermediate nodes to re-randomize a ciphertext, the sender attaches an encryption of value '1' with the recipient's public key. As such, the packet consists of the following tuple:

$$\langle E(1), E(P)\rangle = \langle (g^{r_1}, h^{r_1}), (g^r, P \cdot h^r)\rangle$$

Therefore, without knowledge of the recipient's public key, a node may randomize $E(P)$ as:

$$E(1)^{r_2} \cdot E(P) = (g^r \cdot (g^{r_1})^{r_2}, P \cdot h^r \cdot (h^{r_1})^{r_2})$$

$$= (g^{r+r_1r_2}, P \cdot h^{r+r_1r_2})$$
$$= E(P)$$

Each message consists of multiple ciphertexts, because a single one can only encrypt up to $\log q$ bits (typically 2048 bits) of plaintext. For instance, two ciphertexts are sufficient for delivering short, SMS-style messages. Additionally, when checking an encrypted message for ownership, a node may simply attempt to decrypt $E(1)$. If the output is indeed '1,' the rest of the message is decrypted and displayed to the user.

### E. MESSAGE INTEGRITY

This particular version of the ElGamal cryptosystem is vulnerable to a *message hijacking* attack, due to the multiplicative masking of the plaintext packet $P$. Specifically, an attacker, without knowledge of the actual recipient of a message, can utilize $E(1)$ to hijack the original message and send his own version of the message to that receiver. In particular, given $E(1) = (g^r, h^r)$, the attacker may generate a new message $P'$ as $E(P') = (g^r, P' \cdot h^r)$ and replace the original message $E(P)$. The recipient node is not able to detect this action, because the adversary can compute the correct hash digest to match the new packet contents. By generating his own public key $\mathcal{PK}'$, the attacker can establish a direct communication with the recipient, in order to learn her identity.

To thwart this type of attack, we will leverage the additive version of the ElGamal cryptosystem, where the plaintext message is hidden in the exponent of the public key. More specifically, the ciphertext of a packet $P$ has the form $E(P) = (g^r, h^{P+r})$ and, to allow for ciphertext re-randomization, it is sufficient to attach $E(0) = (g^{r_1}, h^{r_1})$. Note that, the presence of $E(0)$ is of no real use to the adversary, because it can not be employed to produce the encryption of an arbitrary packet $P'$.

Nevertheless, an important limitation of the aforementioned cryptosystem is that the decryption function necessitates a discrete log computation. As such, it can not be used to encrypt arbitrarily large messages. To overcome this limitation, we will use a mixed system where (i) the multiplicative version of the ElGamal cryptosystem is used for message encryption (as before), and (ii) the additive version is used to encrypt a session key $K$ that will turn the hash function in packet $P$ (Section III-C) into an HMAC. Therefore, each plaintext packet $P$ will now have the following form:

$$P = \langle nym, \mathcal{PK}, m, K, HMAC_K(nym|\mathcal{PK}|m)\rangle$$

Similarly, the encrypted version of the packet will consist of the following tuple:

$$\langle E(0|1), E_A(K), E_M(P)\rangle$$

where $E_A(\cdot)$ and $E_M(\cdot)$ represent the additive and multiplicative versions of the ElGamal cryptosystem. Note that, the encryptions of '0' and '1' under the two versions are identical, i.e., both are equal to $(g^{r_1}, h^{r_1})$. As such, we only need to include one randomization ciphertext per message, which we denote as $E(0|1)$. To verify the integrity of a message, the recipient node will (i) decrypt $E_M(P)$ and retrieve $K$, (ii) verify that the HMAC value is correct, and (iii) verify that $E_A(K)$ is indeed an encryption of the retrieved key. We should emphasize that, for the last step, it is not necessary to obtain $K$ from the corresponding ciphertext, which is infeasible due to the discrete log computation. Instead, the recipient will partially decrypt the ciphertext by computing $h^K$, and then verify that the exponent of the computed value is equal to $K$. To launch a successful message hijacking attack, an adversary must now guess the session key $K$ in order to compute the correct value of the HMAC. We include a more detailed discussion of this message hijacking attack in Section V.

## IV. LEVERAGING SYMMETRIC ENCRYPTION

Public key cryptographic operations are computationally expensive, so ciphertext re-randomization will impose a significant overhead on the participating nodes. Therefore, the aforementioned protocol (Algorithm 1) is not suitable for transmitting large messages that are in the order of thousands of bytes in size, but is rather limited to short, SMS-style messages.

To this end, a hybrid solution is more favorable, where the message is encrypted with an efficient symmetric cipher and the session key of the cipher is encrypted with the public key of the recipient. Nevertheless, this approach is not directly applicable to our forwarding algorithm, because symmetric ciphers are inherently deterministic and, thus, ciphertext re-randomization is infeasible. As an alternative solution, we propose to re-encrypt (with a new session key) the symmetric ciphertext at each forwarding step. There are two important issues to resolve here: (i) how do we communicate the underlying session keys to the receiver in order to facilitate the successful removal of the multiple encryption layers, and (ii) during that process, how do intermediate nodes remain oblivious to their position along the path, i.e., how do we render the encryption/decryption order irrelevant.

The straightforward answer to the second question is to use the symmetric cipher in counter mode, i.e., as a *stream* cipher. An example is shown in Figure 3, where the (encrypted) message $m$ is is split into $N$ blocks $(m_1, m_2, \ldots, m_N)$ that match the block size of the underlying symmetric cipher. Each intermediate node $i$ selects a random session key $K_i$ that is used to encrypt a series of $N$ increasing counter values (known to all parties). The output of the cipher for counter $j$ is XORed with $m_j$, in order to produce a re-encrypted value of $m_j$, namely $c_j$. Due to the properties of the XOR operation, the order in which the recipient removes the multiple encryption layers is not important for correct decryption.

Nevertheless, we are still left with the problem of transmitting the encryption keys $K_i$ to the recipient node. A possible solution is for every node to encrypt its session key with the public key of the recipient, using the randomization ciphertext $E(0|1)$ that is included in the message. However, to maintain anonymity, the message size should remain constant every time it passes through a node. In other words, we can not simply add one ciphertext at each forwarding node, because

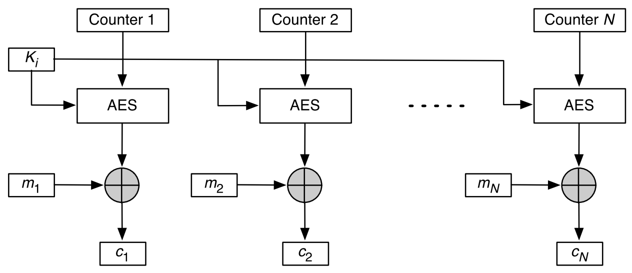


**FIGURE 3.** **Stream cipher.**

an adversary can easily backtrack and identify the sender. The correct approach is to include a vector of $E_M(1)$ values with every message, and then let intermediate nodes choose one to encrypt their session key (this would be done in a round-robin manner, by employing a circular shift of the ciphertexts at each node). While this approach would work in practice, it may potentially incur an overhead that is larger than the one of our basic forwarding protocol. The reason is that, to minimize the collision probability that would render the message undecipherable, the size of the vector containing encryptions of '1' must be equal to the expected worst-case path length (number of hops) between the source and destination nodes. As a result, the forwarding overhead could increase dramatically, because all the public key ciphertexts must be re-randomized at every forwarding step.

To this end, we propose a novel solution, where we combine multiple session keys within each public key ciphertext. Recall that, the ElGamal cryptosystem is multiplicatively homomorphic, which allows us to compute the multiplication of a number of keys within the same ciphertext. Consider, for example, a ciphertext $E_M(K_1)$ that contains the encryption of key $K_1$ (assume the ciphertext has gone through the re-randomization process). The intermediate node can then generate a new session key $K_2$ and combine it with $K_1$, by multiplying it with the second term of the ciphertext:

$$(g^r, K_1 \cdot h^r) = E_M(K_1)$$
$$(g^r, K_2 \cdot K_1 \cdot h^r) = E_M(K_1 \cdot K_2)$$

Obviously, in order for the recipient node to successfully decrypt the anonymous message, the individual session keys $K_i$ should be prime numbers. Furthermore, to defeat brute-force attacks on the generated session keys, the underlying key space should be large enough to match the desired security level, e.g., $2^{128}$. (Note that, the semantic security of the ElGamal cryptosystem protects against the disclosure of the newly added key at every step.) This simple approach can have a significant impact on the computational overhead of the message forwarding process. For instance, if a single ElGamal ciphertext can encode 10 session keys and we expect that the maximum path length on the DTN is 100, then we can transmit all intermediate keys to the recipient with just 10 ciphertexts instead of 100.

A final obstacle that we need to address is the retrieval of the session keys at the destination node. Specifically, given a composite integer $S = K_1 \cdot K_2 \cdots K_n$ that has gone through $n$ intermediate nodes, how do we efficiently retrieve the individual keys? This task involves a factorization algorithm but, unfortunately, there is no *general purpose* algorithm that can factor a composite number efficiently in polynomial time. Instead, our approach is to utilize Pollard's $p-1$ [41] *special purpose* factorization algorithm that can efficiently compute prime factors of a certain type. In particular, Pollard's algorithm can compute prime factors $p$ that are *smooth* with respect to a bound $B$, i.e., all the prime factors of $p-1$ are $\leq B$. Given a smoothness bound $B$, the computational complexity of Pollard's $p-1$ algorithm for computing a single prime factor of a composite integer $S$, is $O(B \cdot \ln S / \ln B)$ modular multiplications [42].

Algorithm 2 illustrates our proposed symmetric key generation algorithm, which outputs *prime* keys that are smooth with respect to a bound $B = p[l]$. Specifically, the algorithm receives as an input the first $l$ primes, and computes a key of the form $2p_1p_2 \cdots p_j + 1$. The individual primes $p_i$ are chosen uniformly at random from the first $l$ primes. This process is repeated until the resulting key is a prime number.

**Algorithm 2** Symmetric Key Generation Algorithm
```
1: procedure KeyGen($p[1..l], j$)
2:     // Input: First $l$ primes; number $j$ of primes in key
3:     while true do
4:         $key \leftarrow 2$;
5:         for $i$ in $[1, j]$ do
6:             $k \xleftarrow{R} [1, l]$;
7:             $key \leftarrow key \cdot p[k]$;
8:         end for
9:         $key \leftarrow key + 1$;
10:        if $key$ is prime then
11:            return $key$;
12:        end if
13:    end while
14: end procedure
```

For security, we should choose appropriate values for $l$ and $j$, such that the underlying key space is sufficiently large. In our implementation, we used the values of $l = 10,000$ and $j = 13$. Under these settings, there are approximately $2^{141}$ unique outputs with an average length of 203 bits. According to the prime number theorem, roughly one in 140 of these numbers are prime, thus giving us sufficient security against brute-force attacks.

In terms of performance, the hybrid approach is superior to the public key one only when the message size is larger than a certain threshold. To this end, the most important performance metric in our system is the computational overhead of the message re-randomization process. Assume that, under the public key method, the message can fit into $N_p$ ciphertexts. Furthermore, assume that the hybrid approach necessitates $N_h$ ciphertexts to communicate the symmetric keys to the recipient node. Given that the overhead of symmetric encryption is negligible compared to the public key re-randomization operations, the hybrid approach is preferable when $N_h < N_p$.

## V. ANONYMITY PROPERTIES

In this section, we discuss in detail the anonymity properties of our messaging network. We consider two types of adversaries, namely passive and active adversaries.

### A. PASSIVE ADVERSARIES

A passive adversary (or eavesdropper) executes the message forwarding protocol correctly, but at the same time has the ability to monitor and collect any transmitted messages within its communication range. The objective is to analyze these communication transcripts in order to identify active sender/receiver pairs. Our forwarding protocol protects against the most powerful passive adversaries, i.e., adversaries that have access to all communications across the entire DTN. The reason is that the attacker is unable to correlate the inbound messages received by a DTN node to the outbound messages transmitted by that node, due to the semantic security of the ElGamal cryptosystem. As a result, it is computationally infeasible for an attacker to trace any message from its source to its destination.

In fact, under the passive adversarial model, our messaging system offers a much stronger notion of anonymity, namely sender and receiver *unobservability*. Here, a powerful eavesdropper, capable of monitoring all network traffic, is unable to observe the act of sending or receiving a message by any node in the system. Unobservability is achieved because of the following two features of our design. First, the fixed buffer space at each node necessitates that newly created messages replace some of the existing messages in the buffer. As such, the message creation process is hidden from the outside world. (Nodes that have recently joined the network should wait until their buffer is full, before they start introducing new messages.) Second, the probabilistic forwarding algorithm dictates that only a fraction $f$ of a node's buffer is sent to a newly discovered node. As a result, if the recipient node removes its message from the buffer, an external observer will be oblivious to this event.

### B. ACTIVE ADVERSARIES

Active attackers are much more powerful than passive ones and have the ability to compromise a fraction of the honest nodes. Once compromised, a node can be controlled remotely by the adversary and follow his instructions. Such malicious nodes can be instructed to launch various types of attacks against honest users. For instance, under a *replay* attack, a malicious node may introduce old messages into the network. Nevertheless, this has no effect on the anonymity of the system, because all messages are re-randomized at each step of the random walk. In addition, replayed messages can be identified at the recipient node, by simply matching their hash digests (HMACs) against the ones that were previously received.

A second realistic attack is for malicious nodes to overwhelm a victim with their own *tagged* messages that are generated with the attacker's public key. Such messages are easily identified by the adversary, even when they are re-randomized by intermediate nodes. After some time, the victim's buffer will not contain any legitimate messages and, if it sends or receives a new message, the adversary can detect that by examining the transmitted messages. However, as long as honest nodes interact with each other frequently (i.e., the fraction of malicious nodes is not large), the anonymity of the system is maintained. Note that, this is not an attack specific to our system, but is applicable to all onion routing based methods as well.

On the other hand, if a legitimate message travels from a source to the destination through a series of malicious nodes, our system will not leak any information to the adversary, due to the random walk nature of the routing mechanism. Indeed, previous work [11], [29] has relied on the concepts of *traceable rate* and *anonymity* to quantify the performance of DTN onion routing algorithms. The traceable rate indicates the percentage of path segments that are disclosed from compromised nodes, while anonymity refers to the degree a message's path is identifiable within an anonymous set. These concepts do not apply to our protocol because, unlike onion routing, even a malicious node will not know what a packet's next destination is. As a result, no partial path is ever disclosed to any node (honest or malicious), which is a significant advantage over onion routing.

Finally, an attack that is unique to our system is the *message hijacking* attack that was briefly mentioned in Section III-E. This attack is made possible by the inclusion of a re-randomization ciphertext within every packet, which can be used to replace the contents of the original packet and transfer ownership of the packet to the attacker node (thus allowing him to communicate with the victim and learn her identity). The utilization of HMACs and additive ElGamal ciphertexts can thwart simple message hijacking attacks, however, a brute-force attack is still possible. In particular, if the attacker wants to know whether Alice is the recipient of *any* anonymous message, he can launch a brute-force attack by hijacking every message that he receives and assuming that its recipient is Alice.

For this attack to be successful, the adversary has to (i) know Alice's public key that is used in the communication, and (ii) receive a message that is destined towards Alice. Therefore, the simplest approach to eliminate such attacks is for the communicating nodes to exchange their keys privately, in an offline phase. If the adversary has no knowledge of Alice's public key, it is infeasible to produce a valid encryption of the HMAC key, i.e., $E_A(K)$, in order to successfully hijack the packet. Furthermore, such a brute-force attack is easily detected by the majority of the network, since numerous nodes will receive packets with incorrect HMACs, due to the vast number of failed message hijacking attempts.

## VI. SIMULATION RESULTS

In this section, we evaluate experimentally the performance of our proposed message forwarding algorithm, using the ONE DTN simulator engine [43]. To simulate the movement of users within the DTN network, we employed two different approaches. First, we used ONE's simulation engine to generate random walks of 1000 pedestrian users over a period of one week. We leveraged ONE's default pedestrian path maps and the default event generation engine, in order to produce the underlying node meetings (synthetic dataset). We also utilized a real-life dataset (GeoLife) that logs the location of 182 users over a period of five years [44]. Specifically, we wrote a Java based program[3] to parse the raw GeoLife data (GPS coordinates) and make it compliant with ONE simulator's requirements. In order to generate node meetings from the 182 users, we set the communication range to 50m in our parsing script, i.e., two users were assumed to be able to pass a message along if their GPS coordinates were $\leq 50$ meters apart. For both datasets, the new message generation rate was kept at one message/minute and the buffer size at each node was set to 1000 messages.

As performance metrics, we measure (i) the message delivery rate, (ii) the end-to-end delay, and (iii) the total number of relayed messages. We compare our method against the baseline epidemic routing (best delivery rate and lowest delay), and Spray and Wait [36] routing (lowest overhead). For the baseline methods, we assume a standard DTN environment without anonymity, and use an infinite buffer to ensure that no messages are dropped. Note that we do not compare against other anonymous DTN routing protocols, such as [9]–[11], [29], because they do not offer the same level of anonymity as our scheme (see Section V).

Fig. 4(a) shows the message delivery rate as a function of the forwarding probability $f$ (for $k = 10$), evaluated over ONE's synthetic dataset. For $f = 0.2$ our method (RW) delivers 88% of the created messages, while for $f = 0.5$ the rate goes up to 92%. Larger values improve the performance even further, but are not recommended due to their impact on anonymity and computational cost. In comparison, epidemic routing (ER) delivers 96% of the messages, while spray and wait (SW) is slightly worse at 89%. Fig. 4(b) illustrates the end-to-end delay for the same experiment. Epidemic routing is the clear winner due to its flooding nature, while our random walk approach outperforms spray and wait when $f \geq 0.2$.

Fig. 5(a) shows the message delivery rate as a function of the forwarding probability $f$ (for $k = 10$) for the GeoLife dataset. Similar to the results reported for the synthetic dataset, our random walk algorithm outperforms spray and wait for $f \geq 0.5$, and delivers over 80% of the created messages. Note that, for this dataset, the upper bound of the delivery rate, as dictated by epidemic routing, is 86%. Regarding the end-to-end delay, Fig. 5(b) illustrates that the random walk approach outperforms spray and wait across all

[3]https://github.com/julianofischer/geolife-extracter

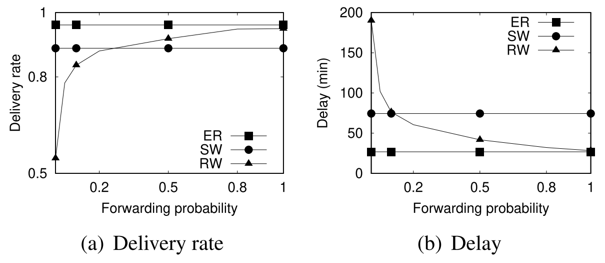


**FIGURE 4. Performance vs. forwarding probability, synthetic dataset.**

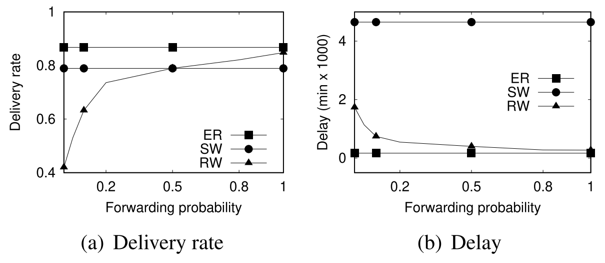


**FIGURE 5. Performance vs. forwarding probability, GeoLife dataset.**

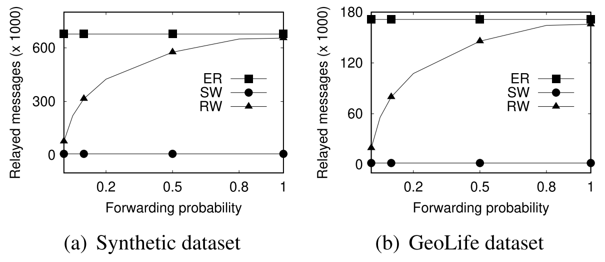
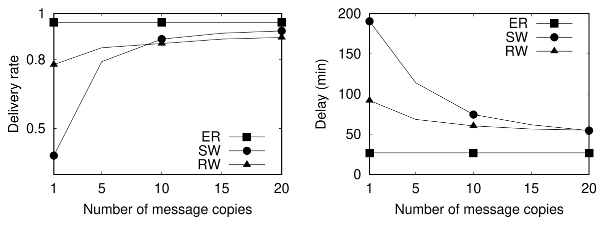


**FIGURE 6. Relayed messages vs. forwarding probability.**

values. Furthermore, the delay is significantly higher compared to the synthetic dataset, due to the limited number of nodes in the network (186 vs. 1000).

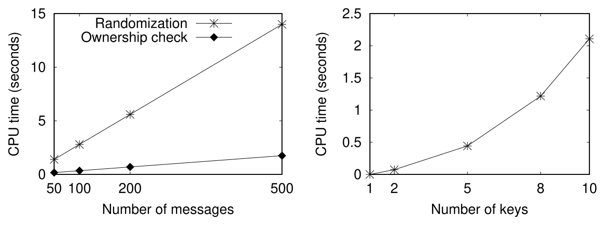
Fig. 6 shows the effect that the forwarding probability has on the number of relayed messages (overhead). For $f \leq 0.2$, the random walk protocol reduces the overhead by 38%-89% compared to epidemic routing (under both datasets). This is very important, because our method necessitates expensive re-randomization operations for each relayed message. Note that, spray and wait has a very low overhead, because it utilizes direct delivery to route messages to their destinations.

Given the benefits of a small forwarding probability $f$ on the system's performance (lower overhead and better anonymity), we next investigate whether it is possible to improve message delivery by generating more copies for new messages. To this end, Fig. 7 depicts the delivery rate and end-to-end delay as a function of the number of message copies $k$, for $f = 0.2$ (synthetic dataset). As expected, increasing the number of copies improves the performance, because it creates more paths that may potentially reach the destination node. (Recall that, in our forwarding algorithm, messages do not include the destination address, so it is easy to miss a delivery even when exchanging messages with the actual



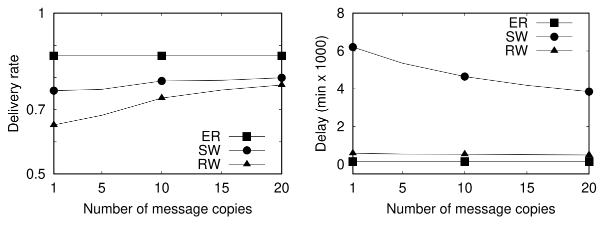
(a) Delivery rate (b) Delay

**FIGURE 7.** Performance vs. $k$ ($f = 0.2$), synthetic dataset.



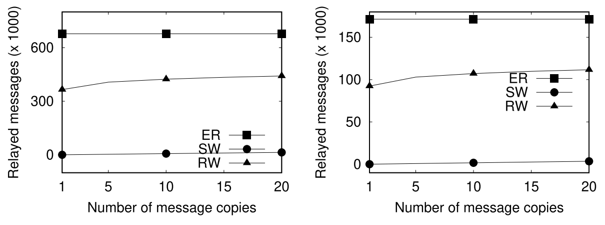
(a) Message forwarding (b) Symmetric key derivation

**FIGURE 10.** CPU overhead.



(a) Delivery rate (b) Delay

**FIGURE 8.** Performance vs. $k$ ($f = 0.2$), GeoLife dataset.



(a) Synthetic dataset (b) GeoLife dataset

**FIGURE 9.** Relayed messages vs. $k$ ($f = 0.2$).

destination node.) For $k = 20$, our algorithm is within 7% of the optimal delivery rate (under epidemic routing), while remaining within a factor of two in terms of end-to-end delay. The number of copies affects spray and wait more drastically, because messages are delivered solely through direct transmission to the destination node.

Fig. 8 shows the results of the same experiment when evaluated with the GeoLife dataset. The same trends can be seen here as well, where, for $k = 20$, the random walk approach is within 10% of the optimal delivery rate, while remaining within a factor of three in terms of end-to-end delay.

Fig. 9 illustrates the number of relayed messages as a function of $k$, for $f = 0.2$ Clearly, increasing $k$ does not have an adverse impact on the network overhead. Using $k = 20$ copies per message is considerably more efficient than epidemic routing, resulting in a 35% lower cost (for both datasets). Spray and wait is again the most efficient approach, because messages are only forwarded upon encountering the destination node.

Next, we investigate the computational overhead of the cryptographic operations that take place during the message forwarding process. We implemented the ElGamal cryptosystem in C, using the GMP library [45] for multiple precision arithmetic. The basic operation involved in the cryptosystem is the modular exponentiation which, for a 2048-bit modulus, took 3.5 ms to complete on a 2.8 GHz Inter Core i7 CPU. Fig. 10(a) shows the CPU cost at the mobile devices as a function of the number of exchanged messages. Ownership check is significantly faster, as it necessitates a single modular exponentiation for each message. On the other hand, message randomization is expensive, because it involves numerous ciphertexts, each requiring two modular exponentiations. Here, we assume that $E_M(P)$ consists of two ciphertexts (512 bytes of plaintext), so there are a total of four ciphertexts in each encrypted packet (see Section III-E). Nevertheless, the overall cost is acceptable, and requires just 5.6 sec of compute time for 200 messages.

Finally, Fig. 10(b) plots the compute time that is required to extract a number of symmetric keys from a single ElGamal ciphertext. It is essentially the cost of running Pollard's $p - 1$ algorithm multiple times, in order to compute the individual keys that are the prime factors of the underlying plaintext. The overall cost on the recipient node is moderate, since we can extract 5 symmetric keys in about 0.5 sec. In our implementation, the size of the plaintext is 2048 bits, which limits the number of keys to 10 per ciphertext. Therefore, depending on the expected maximum path length, we can determine the number of ciphertexts that are required to accommodate the underlying keys. For example, if we set the maximum path length to 50, then we need 5 ciphertexts to encode the keys. In the worst case, this will take about 10 sec at the recipient node to decrypt a message. Note that, the estimate of the maximum path length does not have to be precise, due to the multi-copy nature of our forwarding algorithm. Any message copy that travels through a longer path will simply become undecipherable (Pollard's $p - 1$ algorithm will fail for one or more inputs). On the other hand, provisioning for a path length less than the expected maximum will have a positive effect on the forwarding operation, because of the reduced number of expensive public key randomization operations. In our previous example, using just two ciphertexts will guarantee the successful decryption of any message that goes through, at most, 20 intermediate nodes, while at the same time reducing the randomization cost by 60%.

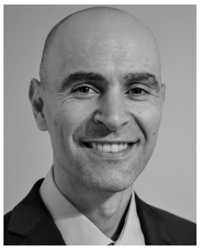
## VII. CONCLUSIONS

Anonymity in private communications has become an important issue for everyday users. To this end, we introduced a novel wireless messaging system with stringent anonymity properties. Our system leverages the opportunistic forwarding mechanism of Delay Tolerant Networks and, as such, it provides stronger anonymity compared to the traditional onion routing paradigm. We proposed two variants of our message forwarding algorithm. The first one is built on public key cryptosystems and is targeted towards short SMS-style messages. The second one is a hybrid system suitable for larger messages, where the message itself is encrypted with an efficient symmetric cipher, while the underlying encryption keys are encrypted with the public key of the recipient node. Our simulations experiments demonstrate that our methods achieve high message delivery rates, while incurring a moderate computational overhead at the mobile devices.

## REFERENCES

[1] New York Times. (2018). *Zuckerberg, Facing Facebook's Worst Crisis Yet, Pledges Better Privacy*. [Online]. Available: https://www.nytimes.com/2018/03/21/technology/facebook-zuckerberg-data-privacy.html
[2] New York Times. (2018). *Equifax Data-Breach Settlement: Get up to 20,000 USD if You Can Prove Harm*. [Online]. Available: https://www.nytimes.com/2019/07/22/business/equifax-data-breach-claim.html
[3] (2015). *AT&T Helped U.S. Spy on Internet on a Vast Scale*. [Online]. Available: https://www.nytimes.com/2015/08/16/us/politics/att-helped-nsa-spy-on-an-array-of-internet-traffic.html?_r=0
[4] *Tor Project: Anonymity Online*. Accessed: Dec. 12, 2019. [Online]. Available: https://www.torproject.org/
[5] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–90, Feb. 1981.
[6] K. R. Fall, "A delay-tolerant network architecture for challenged Internets," in *Proc. ACM SIGCOMM*, 2003, pp. 27–34.
[7] Forbes. (2019). *Hong Kong Protestors Using Mesh Messaging App China Can't Block: Usage Up 3685%*. [Online]. Available: https://www.forbes.com/sites/johnkoetsier/2019/09/02/hong-kong-protestors-using-mesh-messaging-app-china-cant-block-usage-up-3685
[8] *Bridgefy*. Accessed: Dec. 12, 2019. [Online]. Available: https://bridgefy.me/
[9] R. Jansen and R. Beverly, "Toward anonymity in delay tolerant networks: Threshold pivot scheme," in *Proc. MILCOM*, 2010.
[10] C. Shi, X. Luo, P. Traynor, M. H. Ammar, and E. W. Zegura, "ARDEN: Anonymous networking in delay tolerant networks," *Ad Hoc Netw.*, vol. 10, no. 6, pp. 918–930, Aug. 2012.
[11] K. Sakai, M.-T. Sun, W.-S. Ku, and J. Wu, "A framework for anonymous routing in delay tolerant networks," in *Proc. IEEE 25th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2018, pp. 1–10.
[12] S. Bakiras, E. Troja, and X. Xu, "An anonymous messaging system for delay tolerant networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.
[13] J. Camenisch and A. Lysyanskaya, "A formal treatment of onion routing," in *Proc. CRYPTO*, 2005, pp. 169–187.
[14] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a type III anonymous remailer protocol," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2003, pp. 2–15.
[15] G. Danezis and I. Goldberg, "Sphinx: A compact and provably secure mix format," in *Proc. 30th IEEE Symp. Secur. Privacy*, May 2009, pp. 269–282.
[16] B. Möller, "Provably secure public-key encryption for length-preserving chaumian mixes," in *Proc. CT-RSA*, 2003, pp. 244–262.
[17] E. Shimshock, M. Staats, and N. Hopper, "Breaking and provably fixing minx," in *Proc. Int. Symp. Privacy Enhancing Technol. (PETS)*, 2008, pp. 99–114.
[18] L. Zhuang, F. Zhou, B. Y. Zhao, and A. I. T. Rowstron, "Cashmere: Resilient anonymous routing," in *Proc. Symp. Netw. Syst. Design Implement. (NSDI)*, 2005, pp. 301–314.
[19] *Mixmaster*. Accessed: Dec. 12, 2019. [Online]. Available: http://mixmaster.sourceforge.net/
[20] *Mixminion*. Accessed: Dec. 12, 2019. [Online]. Available: http://mixminion.net/
[21] *I2P Anonymous Network*. Accessed: Dec. 12, 2019. [Online]. Available: https://geti2p.net/
[22] Y. Zhang, W. Liu, and W. Lou, "Anonymous communications in mobile ad hoc networks," in *Proc. IEEE 24th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, Mar. 2005, pp. 1940–1951.
[23] J. Kong and X. Hong, "ANODR: Anonymous on demand routing with untraceable routes for mobile ad-hoc networks," in *Proc. 4th ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*, 2003, pp. 291–302.
[24] X. Wu and E. Bertino, "An analysis study on zone-based anonymous communication in mobile ad hoc networks," *IEEE Trans. Dependable Secure Comput.*, vol. 4, no. 4, pp. 252–265, Oct. 2007.
[25] A. Kate, G. M. Zaverucha, and U. Hengartner, "Anonymity and security in delay tolerant networks," in *Proc. Int. Conf. Secur. Privacy Commun. Netw. (SecureComm)*, 2007, pp. 504–513.
[26] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proc. CRYPTO*, 1984, pp. 47–53.
[27] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.
[28] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy (SP)*, Mar. 2007, pp. 321–334.
[29] K. Sakai, M.-T. Sun, W.-S. Ku, J. Wu, and F. S. Alanazi, "An analysis of onion-based anonymous routing for delay tolerant networks," in *Proc. IEEE 36th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2016, pp. 609–618.
[30] X. Lu, P. Hui, D. Towsley, J. Pu, and Z. Xiong, "Anti-localization anonymous routing for delay tolerant network," *Comput. Netw.*, vol. 54, no. 11, pp. 1899–1910, Aug. 2010.
[31] N. Magaia, C. Borrego, P. Pereira, and M. Correia, "PRIVO: A privacy-preserving opportunistic routing protocol for delay tolerant networks," in *Proc. IFIP Netw. Conf. (IFIP Netw.) Workshops*, Jun. 2017, pp. 1–9.
[32] N. Magaia, C. Borrego, P. R. Pereira, and M. Correia, "ePRIVO: An enhanced privacy-preserving opportunistic routing protocol for vehicular delay-tolerant networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11154–11168, Nov. 2018.
[33] Q. Jiang, K. Deng, L. Zhang, and C. Liu, "A privacy-preserving protocol for utility-based routing in DTNs," *Information*, vol. 10, no. 4, p. 128, 2019.
[34] J. Viega, M. Messier, and P. Chandra, *Network Security With OpenSSL*. Newton, MA, USA: O'Reilly Media, 2002.
[35] C. Aguilar-Melchor, J. Barrier, L. Fousse, and M.-O. Killijian, "XPIR: Private information retrieval for everyone," *Proc. Privacy Enhancing Technol.*, vol. 2016, no. 2, pp. 155–174, Apr. 2016.
[36] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Efficient routing in intermittently connected mobile networks: The multiple-copy case," *IEEE/ACM Trans. Netw.*, vol. 16, no. 1, pp. 77–90, Feb. 2008.
[37] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *Mobile Comput. Commun. Rev.*, vol. 7, no. 3, pp. 19–20, 2003.
[38] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "Replication routing in DTNs: A resource allocation approach," *IEEE/ACM Trans. Netw.*, vol. 18, no. 2, pp. 596–609, Apr. 2010.
[39] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Efficient routing in intermittently connected mobile networks: The single-copy case," *IEEE/ACM Trans. Netw.*, vol. 16, no. 1, pp. 63–76, Feb. 2008.
[40] T. El Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Proc. Adv. Cryptol.* Springer, 1985, pp. 10–18.
[41] J. M. Pollard, "Theorems on factorization and primality testing," *Math. Proc. Cambridge Phil. Soc.*, vol. 76, no. 3, pp. 521–528, Nov. 1974.
[42] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, 2001.
[43] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE simulator for DTN protocol evaluation," in *Proc. 2nd Int. ICST Conf. Simul. Tools Techn.*, 2009, p. 55.
[44] Y. Zheng, L. Wang, R. Zhang, X. Xie, and W.-Y. Ma, "GeoLife: Managing and understanding your past life over maps," in *Proc. 9th Int. Conf. Mobile Data Manage. (MDM)*, Apr. 2008, pp. 211–212.
[45] *The GNU MP Bignum Library*. Accessed: Dec. 12, 2019. [Online]. Available: https://gmplib.org/

**SPIRIDON BAKIRAS** (Member, IEEE) received the B.S. degree in electrical and computer engineering from the National Technical University of Athens, in 1993, the M.S. degree in telematics from the University of Surrey, in 1994, and the Ph.D. degree in electrical engineering from the University of Southern California, in 2000. He is currently an Associate Professor with the College of Science and Engineering, Hamad Bin Khalifa University, Qatar. Before that, he held teaching and research positions at Michigan Technological University, The City University of New York, The University of Hong Kong, and The Hong Kong University of Science and Technology. His current research interests include database security and privacy, mobile computing, and spatiotemporal databases. He is a member of the ACM, and a recipient of the U.S. National Science Foundation (NSF) CAREER Award.



**ERALD TROJA** received the Ph.D. degree in computer science from The Graduate Center, CUNY. He is currently an Assistant Professor of computer science with the School of Professional Studies, St. John's University acting as the Program Director for its cybersecurity program. His research interests include security and privacy with a focus on secure and privacy-preserving computations, location privacy and applied cryptography as well as mobile computing with a focus on location-based services, spatial databases, cognitive radio networks, and delay tolerant networks.

**XIAOHUA XU** received the Ph.D. degree in computer science from the Illinois Institute of Technology, Chicago, in 2012. He is currently an Assistant Professor with the Department of Computer Science, Kennesaw State University, Georgia. His teaching and research interests include network security, machine learning, and wireless networking.

**JULIANO FISCHER NAVES** received the B.Sc. degree in computer science from Universidade Federal do Mato Grosso and the M.Sc. and D.Sc. degrees in computer science from Universidade Federal Fluminense. He is currently a Professor with the Instituto Federal de Educação, Ciência e Tecnologia de Rondônia. His major research interests include, but are not limited to, wireless networks, Internet architectures and protocols, network security and delay, and disruption tolerant networks.