An Anonymous Messaging System for Delay Tolerant Networks

Spiridon Bakiras
College of Science and Engineering
Hamad bin Khalifa University
Email: sbakiras@hbku.edu.qa

Erald Troja
The Graduate Center
City University of New York
Email: etroja@graduatecenter.cuny.edu

Xiaohua Xu
Department of Computer Science
Kennesaw State University
Email: xxu6@kennesaw.edu

***Abstract*—Security and anonymity are vital components in today's networked world, and play critical roles in several real-life situations, such as whistleblowing, intelligence operations, oppressive governments, etc. In this paper, we study anonymous communications in the context of Delay Tolerant Networks (DTNs). Existing work in this area relies on the standard onion routing paradigm to provide anonymity and is, therefore, vulnerable to malicious nodes. To this end, we introduce a novel message forwarding algorithm that utilizes random walks to deliver messages to their destinations. By removing the requirement to list all the intermediate nodes on the end-to-end path, our method enhances considerably the anonymity of the underlying communications. Our simulation results show that the proposed forwarding algorithm achieves high message delivery rates, at the expense of a moderate computational overhead at the mobile devices.**

I. Introduction

Technological advances in software vulnerabilities and communications surveillance have turned privacy breaches into a commonplace. Incidents like the Yahoo breach [1] or the widespread surveillance of private citizens' communications by the NSA [2], illustrate that no piece of personal information is safe in the digital world. It is, thus, not surprising that users are becoming increasingly interested in secure and anonymous communications. *Tor* (the onion router) [3] is currently filling up that need, by providing a low-latency platform that anonymizes the user's web browsing activities. Tor is utilized by a diverse user population, including the military, media, activists, and even family and friends.

Tor is inspired by Chaum's work on mix nets [4]. Specifically, it employs the concept of *onion routing* that encapsulates each message into multiple layers of encryption. The message then travels through a series of predetermined nodes (selected by the source) before reaching the destination. Each intermediate node removes one encryption layer, which in turn reveals the next hop towards the destination. The onion routing paradigm is adopted by the majority of anonymous networks that operate today, because of its efficiency and low end-to-end delay. Nevertheless, onion routing is extremely vulnerable to malicious nodes, because they have the potential to reveal part of the route between the communicating parties. In the extreme case, when all intermediate nodes are compromised, the anonymity of the conversation is broken.

Delay Tolerant Networks (DTNs) [5] have also been considered as the underlying platform for building anonymous communication systems. DTNs employ opportunistic routing by bouncing messages randomly among roaming nodes in a store-carry-and-forward manner. As such, DTNs are more resilient against malicious nodes, because they operate in a peer-to-peer (P2P) manner without any infrastructure support. Their location-based forwarding mechanism limits the attack vector of an adversary, and forces malicious nodes to get physically close to the victim in order to launch an attack. DTN-based messaging applications have gained a lot of attention recently, mainly due to the popularity of the *FireChat* app [6], [7].

Unfortunately, existing work on anonymous DTN communications [8], [9] also relies on the concept of onion routing. In particular, nodes are partitioned into groups and the source node selects specific groups that the message has to travel through before reaching the destination. Furthermore, these schemes rely on a trusted key generator to produce the private keys that are shared by the groups, which is an obvious security risk. In this paper, we diverge from the onion routing paradigm and design a novel, decentralized network for anonymous communication. We leverage the opportunistic nature of DTNs to deliver messages with the traditional store-carry-and-forward approach. By removing the requirement to list intermediate nodes/groups on the end-to-end path, our method enhances considerably the anonymity of the underlying communications. Our simulation results show that the proposed forwarding algorithm achieves high message delivery rates, at the expense of a moderate computational overhead at the mobile devices.

The remainder of the paper is organized as follows. Section II reviews previous work on anonymous communications and Section III presents the details of our anonymous messaging system. Section IV discusses the anonymity properties of our design and Section V presents our simulation results. Section VI concludes the paper with some directions for future work.

II. Related Work

There is a plethora of research work on onion routing networks, such as Ref. [10], [11], [12], [13], [14], [15], that mainly differ on the cryptographic constructions of the underlying encryption layers. In addition to theoretical work, several anonymous networks (besides Tor) have been deployed at some point, including Mixmaster [16] and Mixminion [17] for anonymous email, and I2P (Invisible Internet Project) [18] for anonymous messaging, web browsing, blogging, email, and file sharing. However, all the aforementioned protocols are targeted towards networks with a stable/known connectivity.

The first work towards security and anonymity in DTNs is due to Kate et al. [19] that study anonymous Internet access in remote areas. The authors employ identity-based cryptography (IBC) [20] to build an anonymous authentication protocol that allows users to authenticate themselves using pseudonyms instead of real identities. The main limitation of this approach is that it requires a *trusted* key generator, which computes and distributes private keys to the users. This is a single point of failure and, if compromised, the attacker has access to all the private keys, thus breaking the security and anonymity of the entire system.

Jansen and Beverly [8] propose the threshold pivot scheme (TPS), which adapts the onion routing paradigm in the DTN environment. In particular, the authors employ a *group* onion routing algorithm that uses threshold secret sharing [21] to distribute the encryption key among multiple groups. Therefore, in order to reveal the message's destination, the secret has to be reconstructed by nodes in at least τ distinct groups. TPS has several drawbacks. First, it is vulnerable to malicious (or simply colluding) nodes that can decrypt all the receivers' identities once they gain access to τ different groups. Having access to all the groups may also reveal the sender of a message, by simply tracking it as it passes through the different groups. Second, the protocol necessitates that each group maintains a unique public/private key pair, which is extremely challenging to achieve without a trusted third party.

Similar to TPS, ARDEN [9] also employs the group onion routing mechanism to provide anonymity. However, the groups are not statically defined. Instead, the sender leverages attribute-based encryption (ABE) [22] to construct random groups, based on the IDs of nodes that are currently present in the network. Although ARDEN decreases the probability that an adversary has control over nodes in all the generated groups, it still has some major limitations. First, IBE requires a trusted key generator to generate and distribute secret keys to the users. As mentioned previously, this is a single point of failure, and any attacker that gains access to the users' keys can decrypt all traffic in the network. Second, to construct the group onion route, the sender node must have knowledge of the IDs from a large number of users that are currently in the network. This is not a trivial task, and may incur a significant overhead in the system. Finally, ABE is an expensive cryptographic primitive, which imposes a large computational burden on the mobile devices.

III. Anonymous Messaging System

In this section, we present the details of our anonymous messaging system. We start by outlining the adversarial model, and then introduce the various elements of our design, including key management, message forwarding, and cryptographic solutions.

A. Threat Model

Our system will defend against powerful adversaries that may launch both *passive* and *active* attacks. We allow passive adversaries (eavesdroppers) access to all data communications across the entire anonymous network. Naturally, we assume that adversaries run in polynomial time, so they are not able to break the underlying cryptographic protocols. Active adversaries have the ability to compromise a number of honest nodes and turn them into malicious ones. Malicious nodes can attack legitimate users, by forwarding to them carefully crafted messages (including replay messages) and examining the resulting output. The objective is to identify any pair of users that are communicating with each other. Note that, in this work, we do not address denial-of-service (DoS) attacks, since such attacks are not specific to our system (i.e., they are not cryptographic in nature) and can take place in any network environment.

B. Key Management

The first requirement of an anonymous network is data *confidentiality*, i.e., the ability to send messages that can not be deciphered by an eavesdropper. In our system, we employ end-to-end encryption, using the existing public key infrastructure (PKI) [23]. Specifically, we assume that the sender has knowledge of the recipient's public key, and uses this key to encrypt the corresponding message. As a result, only the recipient can successfully decrypt that message. There are several ways for users to retrieve the necessary public keys for communication. For example, the keys could be stored at a public server and be available for download on demand. However, to preserve anonymity, a user must download all keys. Furthermore, all keys should be signed by a trusted certificate authority (CA), in order for users to verify their authenticity. (Note that, in contrast to existing work, all users generate their own public/private key pairs.) Alternatively, users may choose to exchange their public keys privately, without involving a third party. This assumes that the parties know how to exchange their keys in a secure manner.

C. Message Forwarding

The novelty of our work lies in the message forwarding algorithm, which deviates from the traditional onion routing paradigm that is employed in the majority of anonymous networks today. Our solution is to leverage a best-effort approach that forwards messages randomly within the network until they reach their destination (i.e., random walk). However, this is not sufficient to provide anonymity, because an adversary with access to all traffic can trace the entire path between the sender and receiver nodes. Therefore, to preserve anonymity, (i) the identity of the receiver node should be removed and (ii) every message must be re-randomized prior to being forwarded to the next node. In this way, all messages appear as random noise to an adversary, making it infeasible to trace individual messages as they traverse the network. Another requirement is that all messages have the same size, i.e., smaller messages are padded to a default message size, while larger messages are transmitted into multiple chunks.

When Alice wants to send an anonymous message m to Bob, she first constructs a packet[1] P containing the following information ('|' denotes concatenation):

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

$\mathcal{PK}$ is a *fresh* public key that does not exist in the public key database, i.e., it can not be traced back to Alice. This key should be used by Bob, if he wishes to send a reply (or acknowledgment) message to Alice. To hide her real identity, Alice uses a pseudonym (*nym*) instead of her own name[2]. Furthermore, to thwart message modification attacks, she computes and attaches the message digest $H(\cdot)$ of the aforementioned packet data. (Note that the message digest may also be used by Bob to detect duplicate messages.) P is then encrypted with Bob's public key, before being sent out on the network.

DTNs employ a store-carry-and-forward routing algorithm, where nodes store messages in a local buffer B while roaming. Once they discover a new peer (i.e., a node within communication range), they exchange (part of) their buffer contents and continue with their respective trajectories. When a node identifies a message destined to itself, it does not propagate it any further. The simplest forwarding mechanism is flooding, also called *epidemic* routing, where nodes transmit copies of their entire buffer to every node they encounter. The high cost of epidemic routing has triggered the development of numerous efficient protocols, such as spray routing [24], PRoPHET [25], RAPID [26], and many others.

In our context, existing DTN routing protocols face the following challenges: (i) messages do not include the recipient's address or time-to-live (TTL) information, (ii) it is infeasible for a node to identify duplicate messages, and (iii) there is a considerable overhead associated with message forwarding, due to the underlying cryptographic operations. Challenges (ii) and (iii) dictate the use of *multi-copy* routing algorithms, such as Spray and Wait [24], which generate a fixed number k of copies per message. Indeed, without duplicate detection, a flooding algorithm (such as epidemic routing) will generate an enormous amount of messages that go through expensive re-randomization operations at each forwarding step. Challenge (i) also excludes direct transmission routing [27] (i.e., direct delivery of a message to its destination), as well as intelligent algorithms that use node history/location in the forwarding decisions [25].

To tackle the aforementioned challenges, we make the following three design decisions. First, when a node generates a new message, it creates k copies and places them randomly into the output buffer B. Second, to control the flooding process, the buffer size $|B|$ is fixed to a relatively small value, and incoming packets are only accepted if there is sufficient buffer space. Finally, to reduce the computational cost of the packet randomization process (and at the same time improve the anonymity of the system), only a fraction f of the output buffer B is exchanged between two connected nodes. Furthermore, the outbound messages are immediately deleted at the source node. Algorithm 1 illustrates the message forwarding mechanism of our system. Note that, when a node receives a batch of messages from another peer, it has to check all of them for ownership (lines 3–5), because they do not include the destination address. This is done by attempting to decrypt part of the encrypted message, as explained in the following section.

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 project, we will leverage the ElGamal cryptosystem [28] that has this desirable property. The operation of the cryptosystem is summarized below.

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 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

[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 instead.

to the same group, regardless of the underlying public/private key pair.

4) **Decryption**: Compute $m = c_2/c_1^x$.

To enable intermediate nodes to re-randomize a ciphertext, the sender will attach an encryption of value '1' with the recipient's public key. That is, the packet will consist 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:

$$\begin{aligned} E(1)^{r_2} \cdot E(P) &= (g^r \cdot (g^{r_1})^{r_2}, P \cdot h^r \cdot (h^{r_1})^{r_2}) \\ &= (g^{r + r_1 r_2}, P \cdot h^{r + r_1 r_2}) \\ &= E(P) \end{aligned}$$

Each message will consist of multiple ciphertexts, because a single one can encrypt up to $\log q$ bits (typically 2048 bits). As an example, two ciphertexts are sufficient to deliver short, SMS-style messages. In addition, 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. 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 recipient. In particular, given $E(1) = (g^r, h^r)$, the attacker may produce $E(P') = (g^r, P' \cdot h^r)$ and replace the original message $E(P)$. The recipient is unable to detect this replacement, because the attacker can recompute the hash digest to match the new packet contents. By choosing a public key $\mathcal{PK}'$ of his own, the attacker may subsequently try to communicate with the recipient, in order to learn her identity.

To thwart this type of attack, we will leverage another version of the ElGamal cryptosystem (additive ElGamal), where the plaintext message is hidden in the exponent of the public key. 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 useless to the attacker, because it can not be manipulated to produce the encryption of an arbitrary packet P'.

However, a notable 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 hybrid 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:

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

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

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

where $E_A(\cdot)$ and $E_M(\cdot)$ represent the additive and multiplicative versions of the ElGamal cryptosystem. To launch a successful message hijacking attack, an adversary must guess the session key K in order to compute the correct value of the HMAC.

IV. 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.

Passive adversaries A passive adversary (or eavesdropper) does not interfere with the protocol, but instead monitors the underlying communications in order to deduce any piece of relevant information regarding the end-points of an active conversation. In this work, we allow passive adversaries access to all communications. Indeed, even with a global view of the network, an adversary has no advantage in linking an incoming message (entering a node) to an outgoing one (exiting that node). This is due to the semantic security of the ElGamal cryptosystem that renders all messages indistinguishable from each other. Therefore, it is infeasible for an adversary to identify any pair of communicating nodes.

Another important property of our messaging system (under passive adversaries) is sender and receiver *unobservability*. This means that an external observer that sees all traffic can not identify any instance of a node sending or receiving a message. This is due to (i) the fixed buffer space and (ii) the probabilistic forwarding process. More specifically, new messages replace existing ones at the sender's buffer, thus hiding the message creation process. (A node that just joined the system may simply wait until its buffer is full before creating new messages.) On the other hand, by sharing a fraction f of the outbound buffer (instead of the whole buffer), a node may remove a message from the network without leaving any trace.

Active adversaries An active adversary will compromise honest nodes and retrieve all their private keys and gain access to all their internal computations. The compromised (and now malicious) nodes can launch a number of different attacks against honest users. The simplest one is a replay attack, where a malicious node may inject old messages into the network. 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 destination node, by matching their digests against the ones that were received in the past.

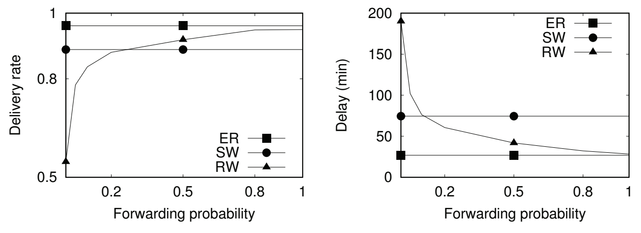
A more serious attack is when malicious nodes flood a victim with their own tagged messages that can be detected even when they are re-randomized. Eventually, the victim's buffer will not contain any legitimate messages and, when it sends or receives a new message, a simple traffic analysis will reveal that. However, as long as honest nodes interact with each other frequently (i.e., the percentage of malicious nodes is not overwhelming), 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.

V. Simulation Results

In this section, we evaluate experimentally the performance of our proposed message forwarding algorithm. We developed the experiments on the ONE DTN simulator [29], where we simulated random walks of 1000 pedestrian users over a period of one week. We utilized ONE's default pedestrian path maps and the default event generation engine, in order to produce the underlying node meetings. The new message generation rate was set to 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 [24] 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 TPS and ARDEN, because they do not offer the same level of anonymity as our scheme (see Section II).

Fig. 1(a) shows the message delivery rate as a function of the forwarding probability f ($k = 10$). 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. 1(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. 2 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. 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.



(a) Delivery rate (b) Delay

Fig. 1. Performance vs. forwarding probability

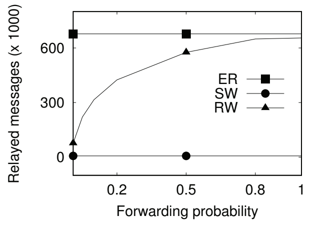
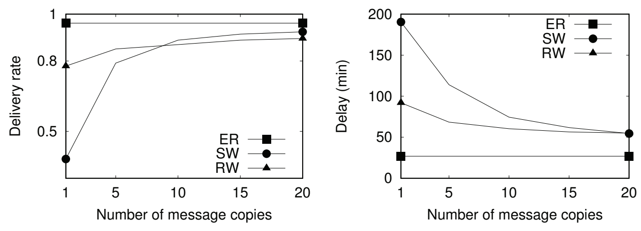


Fig. 2. Relayed messages vs. forwarding probability

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. 3 depicts the delivery rate and end-to-end delay as a function of the number of message copies (k), for $f = 0.2$. 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 destination node.) For $k = 20$, our algorithm is within 7% of the optimal delivery rate (as dictated by the epidemic routing result), 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.



(a) Delivery rate (b) Delay

Fig. 3. Performance vs. k ($f = 0.2$)

Fig. 4 illustrates the number of relayed messages as a function of k, for $f = 0.2$. Clearly, increasing k does not have a 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. Spray and

Wait is again the most efficient approach, because messages are only forwarded upon encountering the destination node.

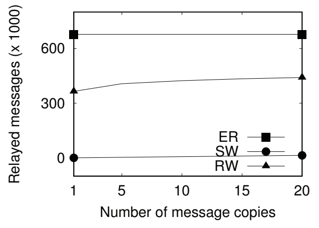


Fig. 4. Relayed messages vs. k ($f = 0.2$)

Finally, 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 [30] 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. 5 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 both $E_A(K)$ and $E_M(P)$ consist of two ciphertexts, so there are a total of six ciphertexts in each encrypted packet (see Section III-E). Nevertheless, the overall cost is acceptable, and requires just 8.4 sec of compute time for 200 messages.

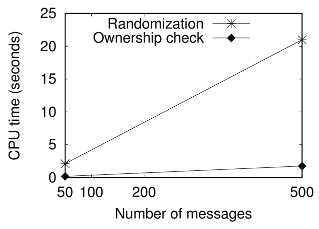


Fig. 5. CPU time vs. number of messages

VI. Conclusions

Anonymity in private communications has become an important issue for everyday users. To this end, we introduce a novel wireless messaging system with stringent anonymity properties. It leverages the opportunistic forwarding mechanism of Delay Tolerant Networks and, as such, it provides stronger anonymity compared to the traditional onion routing paradigm. Our simulations experiments demonstrate that our methods achieve high message delivery rates, while incurring a moderate computational overhead. One shortcoming of our work, is that it utilizes public key encryption for message confidentiality and is, thus, limited to small SMS-style messages. In our future work, we will investigate the feasibility of employing symmetric key encryption in order to allow for larger, multimedia messages.

References

[1] The NY Times. (2016) Defending against hackers took a back seat at Yahoo, insiders say. http://www.nytimes.com/2016/09/29/technology/yahoo-data-breach-hacking.html.
[2] ——. (2015) AT&T helped U.S. spy on internet on a vast scale. http://www.nytimes.com/2015/08/16/us/politics/att-helped-nsa-spy-on-an-array-of-internet-traffic.html?_r=0.
[3] Tor Project: Anonymity Online. https://www.torproject.org/.
[4] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–88, 1981.
[5] K. R. Fall, "A delay-tolerant network architecture for challenged internets," in *ACM SIGCOMM*, 2003, pp. 27–34.
[6] CNN. (2014) FireChat in Hong Kong: How an app tapped its way into the protests. http://www.cnn.com/2014/10/16/tech/mobile/tomorrow-transformed-firechat/.
[7] FireChat. http://opengarden.com/.
[8] R. Jansen and R. Beverly, "Toward anonymity in delay tolerant networks: Threshold pivot scheme," in *MILCOM*, 2010.
[9] C. Shi, X. Luo, P. Traynor, M. H. Ammar, and E. W. Zegura, "ARDEN: anonymous networking in delay tolerant networks," *Ad Hoc Networks*, vol. 10, no. 6, pp. 918–930, 2012.
[10] J. Camenisch and A. Lysyanskaya, "A formal treatment of onion routing," in *CRYPTO*, 2005, pp. 169–187.
[11] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a type III anonymous remailer protocol," in *IEEE Symposium on Security and Privacy (S&P)*, 2003, pp. 2–15.
[12] G. Danezis and I. Goldberg, "Sphinx: A compact and provably secure mix format," in *IEEE Symposium on Security and Privacy (S&P)*, 2009, pp. 269–282.
[13] B. Möller, "Provably secure public-key encryption for length-preserving chaumian mixes," in *CT-RSA*, 2003, pp. 244–262.
[14] E. Shimshock, M. Staats, and N. Hopper, "Breaking and provably fixing minx," in *International Symposium on Privacy Enhancing Technologies (PETS)*, 2008, pp. 99–114.
[15] L. Zhuang, F. Zhou, B. Y. Zhao, and A. I. T. Rowstron, "Cashmere: Resilient anonymous routing," in *Symposium on Networked Systems Design and Implementation (NSDI)*, 2005.
[16] Mixmaster. http://mixmaster.sourceforge.net/.
[17] Mixminion. http://mixminion.net/.
[18] I2P Anonymous Network. https://geti2p.net/.
[19] A. Kate, G. M. Zaverucha, and U. Hengartner, "Anonymity and security in delay tolerant networks," in *International Conference on Security and Privacy in Communication Networks (SecureComm)*, 2007, pp. 504–513.
[20] A. Shamir, "Identity-based cryptosystems and signature schemes," in *CRYPTO*, 1984, pp. 47–53.
[21] ——, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.
[22] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *IEEE Symposium on Security and Privacy (S&P)*, 2007, pp. 321–334.
[23] J. Viega, M. Messier, and P. Chandra, *Network Security with OpenSSL*. O'Reilly Media, 2002.
[24] 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, 2008.
[25] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *Mobile Computing and Communications Review*, vol. 7, no. 3, pp. 19–20, 2003.
[26] 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, 2010.
[27] 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, 2008.
[28] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Advances in Cryptology*, 1985, pp. 10–18.
[29] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *Proc. International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SimuTools)*, 2009, p. 55.
[30] The GNU MP Bignum Library. https://gmplib.org/.