Distributed Privacy-Protecting Routing in DTN: Concealing the Information Indispensable in Routing

Kang Chen
Department of Electrical and Computer Engineering
Southern Illinois University, Carbondale, IL, USA 62901
Email: kchen@siu.edu

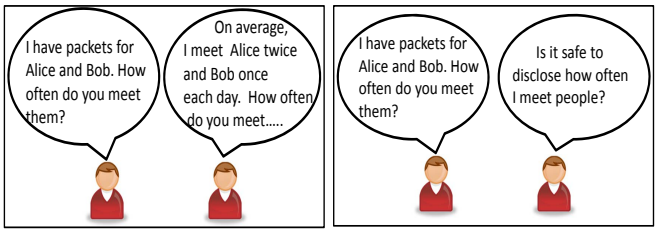
Haiying Shen
Department of Computer Science
University of Virginia, Charlottesville, VA, USA 22904
Email: hs6ms@eservices.virginia.edu

***Abstract*—In state-of-the-art Delay Tolerant Network (DTN) routing algorithms, two encountering nodes rely on routing utilities (e.g., probabilities of meeting other nodes) to decide the better carrier (defined forwarder) for their packets. As the utilities and forwarder information reflect user privacy, nodes may be reluctant to disclose such information, which however is indispensable in routing. To handle this challenge, we propose two distributed strategies to protect the aforementioned private information in utility-based DTN routing algorithms while still guarantying the correctness of packet forwarding, namely meeting Relationship Anonymity (ReHider) and Forwarder Anonymity (FwHider). ReHider anonymizes routing utilities between two encountered nodes, while FwHider additionally hides the forwarder information among the group of encountered nodes on top of the routing utility anonymity. We also present enhanced versions of the two strategies that can better prevent certain malicious behaviors such as probing attack and brute-force attack. The proposed strategies are distributed without the need of a central authority for authentication or key management. They can be applied to any utility-based DTN routing algorithms. Extensive analysis, trace-driven simulation, and smartphone based test demonstrate the effectiveness and energy efficiency of the proposed strategies.**

I. Introduction

In delay tolerant networks (DTNs) [1], due to intermittent node connections, packets are forwarded in a store-carry-forward manner. When two nodes meet (the case with more than two nodes can be derived from pairwise packet exchange naturally), they first compare how likely they can deliver packets carried by them to their destinations (as shown Figure 1(a)). Then, they exchange packets accordingly so that each packet is carried by the node that is more likely to deliver it to the destination, which is defined as the forwarder of the packet in this paper. This process repeats until packets are successfully delivered. In most DTN routing algorithms [2]–[9], such a likelihood is represented by a routing utility, i.e., a node's routing utility for another node represents its probability of delivering packets to the node. For this purpose, the routing utility usually is deduced from node encountering records and/or social properties, e.g., meeting frequency [2]–[4], social closeness [5]–[9], and network centrality [5], [6]. Figure 1(a) shows an example of DTN routing in which the routing utility is represented by meeting frequency.

Such a design rationale means that the routing utility and forwarder information (i.e., which node is the forwarder for which destination) reflect much private information. The former directly shows a node's properties, while the latter can be used to find a node that has a high routing utility for a specific node. Those sensitive information can be exploited by adversaries for harmful attacks. For example, in packet routing [2]–[4], if a malicious node learns the routing utilities of an attack target, it can fabricate utilities larger than those of the attack target to attract and drop its packets. By using the centrality information in the routing utilities in [5], [6], malicious nodes can disseminate viruses more efficiently by first infecting high-centrality nodes.



(a) General DTN routing process. (b) Privacy disclosure concern.

Fig. 1: General packet routing process and privacy disclosure concern.

The development of DTNs consisted of human-operated mobile devices, such as pocket switch networks (PSNs) [5], vehicular delay tolerant networks [10], and rural communication DTNs [11], makes the privacy concern even more significant. In these DTNs, the routing utilities and forwarder information actually reflect people's privacy, such as whom a person often meets and where a person visits frequently, thus leading to much privacy concern from device holders, as shown in Figure 1(b). Therefore, it is essential to protect the routing utility and forwarder information in DTN routing. However, while existing endeavors in this domain are mostly focused on the routing efficiency, few efforts have been devoted to protecting those privacies.

However, concealing such information in DTN routing is non-trivial as it is indispensable for efficient routing. Recall that in the routing process, two encountered nodes compare their utilities to determine forwarders (i.e., best carrier) for their packets and forward packets accordingly (detail in Section III-A). This means that 1) the truthful utility information must be revealed and shared between the two encountered nodes and 2) each node needs to know which node it should forward its packets to. This **paradox** poses a formidable

challenge: *how to anonymize the routing utilities and forwarder information in DTN routing while guaranteeing the correctness of packet forwarding?*

Therefore, in this paper, we propose two distributed strategies to solve the challenge, namely meeting Relationship Anonymity (ReHider) and Forwarder Anonymity (FwHider). The two strategies exploit commutative encryption algorithm [12], order-preserving hash function [13], and a set of novel routing utility exchange and packet forwarding sequences to fulfill the design goal.

ReHider anonymizes routing utilities during the packet routing between two encountered nodes (it can be easily expanded to the case with multiple encountered nodes). The basic idea is to let nodes encrypt utilities before comparing them. However, this leads to two challenges: 1) how to identify encrypted utilities refereeing to the same node for comparison, and 2) how to compare routing utilizes without disclosing their actual values to nodes other than the owner. ReHider uses the commutative encryption algorithm and order-preserving hash function to solve the two problems, respectively. On the basis of ReHider, FwHider further anonymizes the forwarder information during the packet routing process when more than two nodes meet for packet routing. The general idea is to select two nodes to conduct utility comparison (on head node) and packet forwarding (on relay node) separately. Anonymous relaying is adopted in this process to make the two nodes not be able to learn the forwarder information, while packets can be forwarded to their forwarders correctly.

We further design two advanced versions of ReHider and FwHider to better protect nodes from probing attack and brute-force attack. All proposed strategies only need a locally generated encryption key and a hash function on each node and do not assume a central authority for authentication or key management, which is suitable for distributed DTNs.

In summary, the contributions of this paper include:

- We identify the privacy issue in the utility-based DTN routing algorithms. To the best of our knowledge, this is the first investigation on the protection of routing utilities and forwarder information in DTN routing.
- We propose two novel and distributed strategies and their enhanced versions to conceal aforementioned private information in DTN routing without deteriorating the packet routing performance.
- Extensive analysis and experiments demonstrate the effectiveness of proposed strategies.

This paper focuses on utility-based DTN routing algorithms [2]–[9]. The protection of such private information in other types of DTN routing algorithms, e.g., those that require the exchange of context information [14], can be realized in a similar manner and is left to future work.

The remainder of this paper is arranged as follows. Section II introduces related work. Section III presents preliminary background and problem formation. Sections IV and V introduce the two proposed strategies and their advanced versions. Section VI presents the performance evaluation through trace-driven experiments and real deployment on smartphones. Section VII concludes this paper with remarks on future work.

II. Related Work

A. Utility-based DTN Routing

Many utility-based DTN routing algorithms have been proposed with different utility deduction methods [2]–[9]. The works in [2]–[4] deduce the utility from node meeting frequency. PROPHET [2] and MaxProp [3] calculate the utility as the future encountering probability, which is updated upon encountering and ages over time. RAPID [4] also uses previous encountering records to deduce a series of utilities for various routing performance objectives (e.g., minimal average and maximal delay).

Considering that the carriers of mobile nodes in a DTN often can form a social network, several works deduce the routing utility based on social relationships [5]–[9]. The Bubble Rap [5] algorithm considers a community as a group of nodes with frequent contacts, and decides the global ranking for inter-community forwarding and local ranking for intra-community forwarding. SimBet [6] utilizes the network centrality and mobility similarity to deduce the routing utility. In HomeSpread [7], nodes' community visiting preferences are considered to assist content dissemination. In [8] and [9], transient contact patterns and transient community structures are further considered for more accurate packet forwarder selection, respectively.

B. Privacy Protection in DTNs

Many researches have been conducted on protecting various privacies in DTNs or VDTNs [15]–[19]. PreFilter [15] adopts bilinear pairing to ensure that intermediate nodes can only check whether a packet's keywords match the interests of its destination or not, without knowing the actual interests. The work in [16] hides each interest using the solution for the "The Millionaire's Problem" [20], which is designed to compare two items without disclosing their actual values.

EnPass [18] realizes anonymous routing. It specifies the group sequence that a packet should be forwarded through and encrypts the packet with the public key of each group in the same sequence. Then, the packet can be decrypted by each group to discover the next group without knowing its source and destination. ALERT [19] dynamically partitions the network field into zones and randomly chooses a node in each zone to form a non-traceable anonymous route. The anonymous routing helps protect node privacy but generally relies on the concept of "source routing", which may not be efficient in the context of DTNs.

STAP [17] caches packets for a node on places it frequently visits. Then, others do not need to know the node's exact location to send packets to it. In the work of [21], aggregate statistics about sensing results are obtained without exposing individual's sensed data by using additive homomorphic encryption and a novel key management scheme. STAMP [22] lets each node anonymously provide the location proof for co-location nodes to protect node privacy.

Though these methods are effective in protecting different types of privacy in DTNs, no previous work has been proposed to protect node privacy in routing utilities and selected packet forwarders. Our work is the first to protect such indispensable information in DTN routing while still guarantee the correct operation of the routing algorithms.

There are potential other ways of protecting such information. For example, we can only allow nodes to exchange such information among trusted nodes [23]. However, we focus on anonymizing those information at all in this paper because this would provide more complete protection.

III. Preliminaries

A. Network Model and Utility-based Packet Routing

We assume a DTN with N mobile nodes denoted by n_i ($i \in [1, N]$). In this paper, we focus on utility-based routing algorithms, which represent the state-of-the-art DTN routing algorithms. In these algorithms, two encountering nodes first exchange packets that take the other node as destination. Then, they exchange the routing utilities for the destinations of all packets on both nodes. Finally, packets are forwarded to the node that has higher routing utilities for their destinations. We use U_{ij} to denote node n_i's routing utility for n_j:

$$U_{ij} = \{n_i, n_j, v_{ij}\}, \tag{1}$$

where n_i, n_j, and v_{ij} denote the source, target, and value of U_{ij}, respectively. We use utility and routing utility interchangeably in this paper. For simplicity, when we say a node's routing utility for a packet, we are referring to the node's routing utility for the packet's destination.

B. Problem Formation

We first formalize the problem solved in this paper, which includes design goal and adversary model.

1) Design Goal: In this paper, we protect two types of privacy information in utility-based DTN routing algorithms while ensuring correct packet forwarding. By protection, we mean that each node only know the information of itself and cannot know that of other nodes.

- Routing Utility: During the packet routing, each node cannot know the routing utilities of other nodes.
- Forwarder Information: During the packet routing, each node only knows the forwarder information of itself but cannot know that of others (i.e., only knows it is the forwarder for which packets and cannot know such information of other nodes).

2) Adversary Model: Unlike previous works that investigate the security, incentive, or bundle authentication in DTNs [24], [25], we focus on the protection of routing utilities and forwarder information in utility-based DTN routing. We assume an adversary model in which malicious nodes try to learn the aforementioned information of others through the following attacks.

- Eavesdropping: In this attack, a malicious node eavesdrops the communication between other nodes.
- Probing attack: In this attack, a malicious node probes another node's routing utility by repetitively conducting packet routing with the node. After each routing session, the malicious node adjusts its routing utility values based on the comparison result, thus gradually deducing the node's routing utilities.
- Brute-force attack: In this attack, a malicious node collect pairs of clear text and cipher text generated by the encryption algorithm of another node and use such information to break the encryption algorithm.

Those attacks are mainly thwarted in the advanced versions of the proposed strategies. We do **not** consider the collusion attack in this paper and leave it to future work.

C. Cryptographic Techniques

We first introduce two cryptographic techniques that are used in our proposed strategies: commutative encryption algorithm [12] and order-preserving hash function [13].

1) Commutative Encryption: A commutative encryption algorithm $\mathcal{E}(\cdot)$ satisfies the properties below for any keys k_1 and k_2, message M, rational number s and $\epsilon < 1/2^s$

- $\mathcal{E}_{k_1}(\mathcal{E}_{k_2}(M)) = \mathcal{E}_{k_2}(\mathcal{E}_{k_1}(M))$
- $\forall\ M_1 \neq M_2$, $Pr(\mathcal{E}_{k_1}(\mathcal{E}_{k_2}(M_1)) = \mathcal{E}_{k_2}(\mathcal{E}_{k_1}(M_2))) < \epsilon$,

where $\mathcal{E}_{k_1}(M)$ means encrypting message M with key k_1.

Many commutative encryption algorithms exist with different security and complexity levels, such as RSA [26] and one-time pad [27]. The work in [12] proposes to use Pohlig-Hellman encryption to realize a commutative encryption algorithm with acceptable security and complexity.

2) Order-preserving Hashing: An order-preserving hash function $\mathcal{H}(\cdot)$ satisfies properties below for v_1 and v_2 [13]

- If $\mathcal{H}(v_1) = \mathcal{H}(v_2)$, $v_1 = v_2$
- If $v_1 > v_2$, $\mathcal{H}(v_1) > \mathcal{H}(v_2)$.

The order-preserving hashing can compare two items without disclosing their actual values.

D. System Setup

When a system starts, a commutative encryption algorithm $\mathcal{E}(\cdot)$ is selected for all nodes based on the system requirement on security and complexity level. When a node, say n_i, joins in the system, it selects an order-preserving hash function $\mathcal{H}_i(\cdot)$. Except for these steps, no other configurations are needed to support the strategies proposed in this paper.

A node's encryption algorithm, encryption key, and hashing function can change upon each packet routing, thus enhancing the security level of the system.

IV. Meeting Relationship Anonymity

Since DTN routing algorithms are all developed based on pairwise node encountering, we focus on the routing utility protection when two nodes meet for packet routing in this section. Actually, the solution for pairwise encountering can be easily expanded to the scenario when multiple nodes meet for packet routing, as discussed in Section IV-D.

In the following, we explain our strategies using the case when n_1 and n_2 meet for packet routing. Following the packet

routing process introduced in Section III-A, the two nodes first deliver packets destined to n_1 or n_2. They then compare their routing utilities for the destinations of all remaining packets on them, which we assume are $\{n_a, n_b, n_c\}$ ($a, b, c \in [3, N]$, packets for n_1 and n_2 have already been delivered in the first step), to determine their forwarders. We let x denote an element in set $\{a, b, c\}$, i.e., $x \in \{a, b, c\}$. Such a setting is an example and our strategies can be applied to cases with different numbers of destinations.

A. Baseline Meeting Relationship Anonymity (B-ReHider)

B-ReHider realizes anonymous routing utility comparison between two encountered nodes. The rationale in B-ReHider comes from our observation on the structure of the routing utility (shown in Formula 1): a routing utility is disclosed only when both its target and value are disclosed. This means that if only the target or the value is disclosed, others still cannot get any meaningful information. For example, if n_1 tells n_2 that my routing utility for $\mathcal{E}(n_a)$ is v_{1a}, in which the target (i.e., n_a) is encrypted by an encryption function $\mathcal{E}()$. In this case, n_1's routing utility for n_a is still safe.

We exploit this property together with the commutative encryption and the order-preserving hashing function to ensure both anonymous and correct routing utility comparison.

1) Design of B-ReHider:

(a) **Initial Setup:** When two nodes, say n_1 and n_2, meet for packet routing, each node first creates an encryption key, say k_1 and k_2. The two nodes select a node from them as the *comparison node* following a certain rule (i.e., randomly or the one with a larger ID). We suppose n_1 is selected as the comparison node. They then compare their routing utilities for $\{n_a, n_b, n_c\}$ to determine the packet forwarder.

(b) **Utility Encryption:** Routing utilities need to be encrypted to ensure the anonymity during the comparison. Each node first encrypts the targets of its utilities with its key. Beside, n_2 also hashes the values of its utilities in order to hide this information from n_1. After this, each node sends all encrypted utilities to the other node.

$$n_1 \rightarrow n_2 : \mathcal{U'}_{1x} : \{n_1, \mathcal{E}_{k_1}(n_x), v_{1x}\}$$
$$n_2 \rightarrow n_1 : \mathcal{U'}_{2x} : \{n_2, \mathcal{E}_{k_2}(n_x), \mathcal{H}_2(v_{2x})\}$$

After this step, each node only knows the other node's routing utilities with an encrypted target, i.e., $\mathcal{E}_{k_1}(n_x)$ and $\mathcal{E}_{k_2}(n_x)$. Then, even if the values, i.e. v_{1x}, are disclosed, the routing utility anonymity is still kept. However, there maybe a problem when two nodes only need to compare a few routing utilities. This case is discussed later in Section IV-C.

n_1 and n_2 further encrypt the target of all received utilities with their keys. n_2 also hashes the values of received utilities with its hash function. As a result, n_1 has $\mathcal{U''}_{2x} : \{n_2, \mathcal{E}_{k_1}(\mathcal{E}_{k_2}(n_x)), \mathcal{H}_2(v_{2x})\}$, and n_2 has $\mathcal{U''}_{1x} : \{n_1, \mathcal{E}_{k_2}(\mathcal{E}_{k_1}(n_x)), \mathcal{H}_2(v_{1x})\}$. Finally, n_2 sends the encrypted n_1's utilities to the comparison node n_1 for comparison.

$$n_2 \rightarrow n_1 : \mathcal{U''}_{1x} : \{n_1, \mathcal{E}_{k_2}(\mathcal{E}_{k_1}(n_x)), \mathcal{H}_2(v_{1x})\}$$
$$n_1 \text{ has} : \mathcal{U''}_{2x} : \{n_2, \mathcal{E}_{k_1}(\mathcal{E}_{k_2}(n_x)), \mathcal{H}_2(v_{2x})\} \quad and \quad \mathcal{U''}_{1x}$$

(c) **Utility Comparison:** n_1 compares $\mathcal{U''}_{2x}$ and $\mathcal{U''}_{1x}$ to decide the packet forwarder for each destination. The challenge here is to identity the utilities with the same target (i.e., for the same destination) after the encryption. This problem is solved by the property of the commutative encryption. That is, if $\mathcal{E}_{k_1}(\mathcal{E}_{k_2}(n_x)) = \mathcal{E}_{k_2}(\mathcal{E}_{k_1}(n_y))$, we can conclude that $n_x = n_y$. This means that routing utilities for the same target in $\mathcal{U''}_{2x}$ and $\mathcal{U''}_{1x}$ have the same encrypted target and can be easily identified. Further, since all utility values in $\mathcal{U''}_{2x}$ and $\mathcal{U''}_{1x}$ are hashed by the same order-preserving hashing function, i.e., $\mathcal{H}_2()$, the comparison correctness is ensured.

(d) **Decrypting the Comparison Result:** The comparison in the previous step determines which node (n_1 or n_2) is the forwarder for each encrypted destination, e.g., $\mathcal{E}_{k_1}(\mathcal{E}_{k_2}(n_x))$. Then, we need to decrypt them so that each node can know it is the forwarder for which destinations. Suppose that after the comparison, n_1 has higher utility for destination $\mathcal{E}_{k_1}(\mathcal{E}_{k_2}(n_a))$. n_1 first decrypts it with k_1 and obtains $\mathcal{E}_{k_2}(n_a)$. It further sends $\mathcal{E}_{k_2}(n_a)$ to n_2 and tells n_2 that it is the forwarder for this destination. n_2 decrypts the received destination with k_2 and learns that n_1 is the forwarder for destination n_a. n_2 then knows that it is the forwarder for remaining destinations, i.e., n_b and n_c. n_2 also sends this information to n_1 to let it know that it is the forwarder for n_a.

The decrypted comparison result cannot help two nodes deduce the utility values in the encrypted routing utilities. For n_1, even though it can find n_2's encrypted utility for n_a since it is the only one that comes from n_2 and is smaller than the corresponding one from n_1. However, since the routing utility values received by n_1 is hashed by $\mathcal{H}_2()$ (in step ***(b)***), n_1 cannot know the actual utility value. For n_2, since it does not do the comparison, it cannot identify n_1's routing utility for n_a from received routing utilities.

(e) **Packet Forwarding:** Finally, each node forwards packets taking the other node as the forwarder to the other node.

2) Privacy Protection Analysis: In this section, we analyze B-ReHider's capability to anonymize routing utilities and resist attacks mentioned in Section III-B2.

Anonymize Routing Utilities: We first summarize the information that a node can collect in B-ReHider in Table I to analyze whether routing utilities are anonymized.

TABLE I: Information collected by in each node in B-ReHider.

Node	Information
n_1	$\mathcal{U'}_{1x} : \{\mathcal{E}_{k_1}(n_x), v_{1x}, n_1\}$ $\mathcal{U''}_{1x} : \{\mathcal{E}_{k_2}(\mathcal{E}_{k_1}(n_x)), \mathcal{H}_2(v_{1x}), n_1\}$ $\mathcal{U'}_{2x} : \{\mathcal{E}_{k_2}(n_x), \mathcal{H}_2(v_{2x}), n_2\}$ $\mathcal{U''}_{2x} : \{\mathcal{E}_{k_1}(\mathcal{E}_{k_2}(n_x)), \mathcal{H}_2(v_{2x}), n_2\}$
n_2	$\mathcal{U'}_{2x} : \{\mathcal{E}_{k_2}(n_x), \mathcal{H}_2(v_{2x}), n_2\}$ $\mathcal{U'}_{1x} : \{\mathcal{E}_{k_1}(n_x), v_{1x}, n_1\}$ $\mathcal{U''}_{1x} : \{\mathcal{E}_{k_2}(\mathcal{E}_{k_1}(n_x)), \mathcal{H}_2(v_{1x}), n_1\}$

We see from the table that each node can only get the utilities with encrypted targets and/or hashed values. This means each node's routing utilities are anonymized against the other node during the packet routing in B-ReHider.

Eavesdropping: By examining the utilities transmitted in B-ReHider, we find that they cannot be understood by any

eavesdropper because for each transmitted utility, its target is encrypted, i.e., $\mathcal{E}_{k_1}(n_x)$ or $\mathcal{E}_{k_1}(\mathcal{E}_{k_2}(n_x))$, and its utility values are hashed. Therefore, eavesdroppers cannot obtain any meaningful information without knowing k_1, k_2, and the hash functions ($H_1()$ and $H_2()$). This means that B-ReHider can effectively thwart the eavesdropping attack.

Probing Attack and Brute-Force Attack: B-ReHider cannot resist the probing attack and the brute-force attack. First, since the utility comparison result is shared between the two nodes in B-ReHider, a malicious node can easily probe another node's routing utilities by repetitively conducting packet routing (i.e., comparing routing utilities) with it. After each packet routing, the malicious node can adjust its routing utility values based on the comparison result. Then, after several rounds, the node's routing utility values can be gradually deduced.

Second, by examining Table I, we find that n_1 can easily access multiple clear-text and cipher-text pairs of $\mathcal{E}_{k_2}()$ and $\mathcal{H}_2()$. In detail, n_1 can sort $\mathcal{U'}_{1x} : \{n_1, \mathcal{E}_{k_1}(n_x), v_{1x}\}$ by v_{1x} and $\mathcal{U''}_{1x} : \{n_1, \mathcal{E}_{k_2}(\mathcal{E}_{k_1}(n_x)), \mathcal{H}_2(v_{1x})\}$ by $\mathcal{H}_2(v_{1x})$. Since $\mathcal{H}_2()$ is order-reserving, $\mathcal{H}_2(v_{1x})$ has the same order as v_{1x}. As a result, $\mathcal{E}_{k_1}(n_x)$ and $\mathcal{E}_{k_2}(\mathcal{E}_{k_1}(n_x))$ appear on the same position in each sorted set. This means that n_1 can get multiple clear-text and cipher-text pairs: $< \mathcal{E}_{k_1}(n_x), \mathcal{E}_{k_2}(\mathcal{E}_{k_1}(n_x)) >$ and $< v_{1x}, \mathcal{H}_2(v_{1x}) >$ to break $\mathcal{E}_{k_2}(\cdot)$ and $\mathcal{H}_2(\cdot)$.

We thereby propose an enhanced version, named E-ReHider, to better prevent the two attacks in Section IV-B.

3) Summary: In B-ReHider, two encountered nodes compare their routing utilities anonymously. B-ReHider can also thwart the eavesdropping attack. However, B-ReHider cannot resist the probing attack and the brute-force attack, which are solved in the advanced version (E-ReHider) in Section IV-B.

B. Enhanced Relationship Anonymity (E-ReHider)

We further design E-ReHider on the basis of B-ReHider to better prevent the probing attack and the brute-force attack.

1) Preventing the Probing Attack: The probing attack works only when two requirements are satisfied: 1) know the utility comparison result and 2) can repetitively identify the victim node for probing. While the first one is necessary to ensure the packet routing, we limit the second requirement to prevent such an attack. Specifically, we let nodes 1) use a pseudonym to communication with the encountered node for packet routing and 2) change the pseudonym after conducting the packet routing. As a result, a node presents a different pseudonym each time when it meets a node. We call such a feature neighborhood anonymity. The work in [28] has proposed a scheme to realize this feature in DTNs without affecting normal packet routing. It can be directly adopted in E-ReHider. Actually, any techniques that can realize the feature can be integrated into E-ReHider for this purpose.

2) Preventing the Brute-force Attack: As introduced in Section IV-A2, B-ReHider suffers from the brute-force attack mainly because v_{1x} and $\mathcal{H}_2(v_{1x})$ on n_2 have the same order. We then solve the problem by breaking such a property. The general idea is to create zombie destinations, which do not exist in packets on both nodes, and let n_2 modify utilities for those destinations received from n_1, thereby changing the order. We name the utilities for zombie and real destinations as *zombie utilities* and *real utilities*, respectively.

In the following, we introduce key changes in each step of B-ReHider that help better prevent the brute-force attack.

(a)* Initial Setup:** After the initial setup in B-ReHider (introduced in Section IV-A1 step ***(a)), n_1 randomly creates a set of zombie destinations (that are not duplicate with real destinations). We assume the zombie destinations are $\{n_d, n_e, n_f\}$, where $d, e, f \in [3, N] \setminus \{a, b, c\}$. n_1 also informs n_2 the selected zombie destinations. We use z to represent a member in $\{d, e, f\}$, i.e., $z \in \{d, e, f\}$.

(b)* Utility Encryption:** As in the utility encryption step in B-ReHider (introduced Section IV-A1 step ***(b)), two nodes encrypt the targets of their real utilities and zombie utilities and send them to the other node. However, the two types of utilities are sent separately. However, different from the B-ReHider, after receiving the zombies utilities from n_1, n_2 arbitrarily modifies their values from v_{1z} to $\widetilde{v_{1z}}$, as shown below.

$$n_2 : \mathcal{U'}_{1z} : \{n_1, \mathcal{E}_{k_1}(n_z), v_{1z}\} \Longrightarrow \widetilde{\mathcal{U'}_{1z}} : \{n_1, \mathcal{E}_{k_1}(n_z), \widetilde{v_{1z}}\}$$

After this step, n_1 and n_2 mix the zombie utilities and real utilities. Then, both nodes follow the procedure as in the utility encryption step in B-ReHider (Section IV-A1 step ***(b)***) to further encrypt the received utilities. Finally, we have

$$n_2 \rightarrow n_1 : \{\mathcal{U''}_{1x} \cup \widetilde{\mathcal{U''}_{1z}}\}$$
$$n_1 \text{ has} : \{\mathcal{U''}_{2x} \cup \mathcal{U''}_{2z}\} \ \ and \ \ \{\mathcal{U''}_{1x} \cup \widetilde{\mathcal{U''}_{1z}}\}$$

where $\cup$ denotes mixing the two groups.

Above changes can effectively prevent the brute-force attack. Specifically, in step ***(c)***, n_2 modifies the values of n_1's zombie utilities and mixes them with the real utilities before sending them to n_1. Then, $\{v_{1x}, v_{1z}\}$ and $\{\mathcal{H}_2(v_{1x}), \mathcal{H}_2(\widetilde{v_{1z}})\}$ do not have the same order since $\mathcal{H}_2(\widetilde{v_{1z}})$ is different from $\mathcal{H}_2(v_{1z})$. Consequently, n_1 cannot correlate $\mathcal{E}_{k_1}(n_x)$ with $\mathcal{E}_{k_2}(\mathcal{E}_{k_1}(n_x))$ or v_{1z} with $\mathcal{H}_2(v_{1z})$, i.e., cannot easily collect pairs of clear-text and cipher-text for the brute-force attack.

Moreover, adding zombie utilities does not affect the correctness of packet forwarding for two reasons: 1) the comparison between real utilities is not changed since their values are not modified. 2) there are actually no packets destined to zombie destinations on both nodes.

3) Summary: With the above design, E-FwHider can effectively thwart all attacks mentioned in Section III-B2.

C. Preventing a Special Case

When there are only a few utilities to compare, nodes can easily deduce the targets in encrypted utilities. For example, suppose two nodes only need to compare the utility for one node, say n_a. Then, upon receiving the utility from the other node, each node directly knows that its target is n_a. To solve this problem, we require that each node must report at least $M > 5$ destinations in step ***(a)***. If a node has less than M destinations, it arbitrarily selects some zombie destinations. These destinations are processed in the same way as normal

destinations. Then, a node cannot guess utilities directly. Since there actually no packets for these zombie destinations, the packet routing process is not affected and remains correct.

This strategy is applied to all strategies proposed in the paper to prevent privacy leakage in such an extreme case. We do not mention it explicitly in other parts to save space.

D. Expanding to Multiple Nodes

Both B-ReHider and E-ReHider can be easily expanded for the case when multiple nodes meet for packet routing. One simple way is to view the group of encountered nodes as multiple pairwise encountering and conduct packet routing for each pair of nodes (with B-ReHider/E-ReHider applied) sequentially. However, this method has a high cost since it incurs $N_g * (N_g - 1)/2$ pairwise encountering, where N_g is the size of the group. A better way is to select two nodes as the proxy for utility comparison, in which B-ReHider/E-ReHider can be used to anonymize routing utilizes. Such a method is adopted in FwHider and is introduced later in section V.

V. Forwarder Anonymity

ReHider can effectively protect routing utilities. However, it does not anonymize the forwarder information between the two encountered nodes. The forwarder for a destination is the node that has the highest utility value for the destination among all encountering nodes. Such information can be exploited to find a node that has a high routing utility for a specific destination by tracking packets destined to the destination. We then propose a forwarder anonymity method, called FwHider, to further protect the packet forwarder information. FwHider **only** applies to the scenario when a group of nodes (i.e., > 2) meet for packet routing. This is because when there are only two nodes, each node anyway knows the forwarder of each packet, i.e., either itself or the other node.

Without loss of generality, we discuss the scenario when a group of nodes n_1, n_2, n_3, and n_4 meet for packet routing. Following the packet routing process introduced in Section III-A, these nodes first deliver packets destined to n_1, n_2, n_3 or n_4. They then need to compare their routing utilities for the destinations of all remaining packets on them, which we assume are $\{n_a, n_b, n_c, n_d\}$ $(a, b, c, d \in [5, N])$, to determine packet forwarders. We use x to represent an element in $\{a, b, c, d\}$, i.e., $x \in \{a, b, c, d\}$ in this section. Note that the above case if just an example, and FwHider applies to any group size ($>$2) and any number of destinations.

A. Baseline Forwarder Anonymity (B-FwHider)

The rationale in B-FwHider comes from our observation that when there are multiple nodes, we can indirectly relay packets to their forwarders through a relay node. We also ensure that the relay node only knows the forwarder information in which the destination node ID is encrypted (Table II). Therefore, the forwarder information not disclosed.

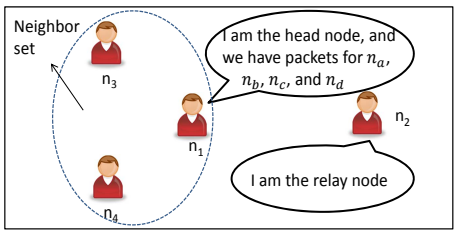


Fig. 2: Illustration of the FwHider scenario.

1) Design of B-FwHider:

***(a)* Initial Setup:** The group of encountering nodes first select the relay node, the neighbor set (defined in the next sentence), and the head node of the neighbor set following a certain rule (e.g., based on node capability). The neighbor set includes all nodes except the relay node. Since the relay node and the head node take certain additional responsibilities, they are selected from nodes that are more likely to stay in current place to ensure their availability. We assume n_2 is the relay node and n_1 is the head node, as shown in Figure 2. The head node then decides a group key k_n and share it with nodes in the neighbor set (i.e., n_3 and n_4). n_2 also generates a key k_2 for anonymous utility comparison and keep it secretly.

***(b)* Utility Encryption:** Each node in the neighbor set encrypts the targets of its utilities with key k_n and sends all encrypted utilities to relay node n_2.

$$n_1 \rightarrow n_2 : \mathcal{U}'_{1x} : \{n_1, \mathcal{E}_{k_n}(n_x), v_{1x}\}$$
$$n_3 \rightarrow n_2 : \mathcal{U}'_{3x} : \{n_3, \mathcal{E}_{k_n}(n_x), v_{3x}\}$$
$$n_4 \rightarrow n_2 : \mathcal{U}'_{4x} : \{n_4, \mathcal{E}_{k_n}(n_x), v_{4x}\}$$

In this process, since the target of each utility is encrypted, the utility anonymity is still kept, i.e., n_2 cannot deduce another node's utility value for any node. After this, n_2 has multiple utilities with the same encrypted target ($\mathcal{E}_{k_n}(n_x)$), i.e., one from each node in the neighbor set. Among utilities with the same encrypted target, n_2 selects the one with the largest value. We denote all selected utilities as neighbor set utilities. Basically, the neighbor set utilities include the maximal utility for each destination in the neighbor set.

(c)* Utility Comparison:** After this, n_1 and n_2 can compare the neighbor set utilities with n_2's utilities to find the maximal utility for each destination in the whole set of encountered nodes. We follow the same procedure as in steps ***(b) and ***(c)*** in B-ReHider (Section IV-A1) to fulfill this task while keeping the anonymity of routing utilities.

TABLE II: An example of the relay table on n_2.

k_n-encrypted destination	Forwarder
$\mathcal{E}_{k_n}(n_a)$	n_1
$\mathcal{E}_{k_n}(n_c)$	n_3
$\mathcal{E}_{k_n}(n_d)$	n_4

***(d)* Creating Relay Table:** After the comparison, n_1 does not decrypt the encrypted destinations (i.e., $\mathcal{E}_{k_2}(\mathcal{E}_{k_n}())$) that take nodes in the neighbor set as forwarders but send such information to n_2 directly. n_2 then decrypts those destinations with its key (k_2) and get the destinations encrypted by k_n

only, which are denoted as k_n-encrypted destinations. Finally, n_2 creates a relay table to record the forwarder for each k_n-encrypted destination, as shown in Table II.

***(e)* Packet Relay:** Packets then are anonymously relayed to their forwarders by the relay node n_2. Specifically, each node in the neighbor set encrypts the destinations of their packets with k_n and sends them to n_2. Note that n_2 also sends all of its packets to n_1 for encryption as it does not know k_n. For each received packet, n_2 searches its k_n-encrypted destination in the relay table. If there is a match, the packet is forwarded to the corresponding forwarder. Otherwise, n_2 keeps the packet since this means that it is the forwarder for the packet. In summary, n_2 does the following if the forwarding table is as shown in Table II.

$$n_2 \rightarrow n_1 : \text{packets with destination } \mathcal{E}_{k_n}(n_a)$$
$$n_2 \rightarrow n_3 : \text{packets with destination } \mathcal{E}_{k_n}(n_b)$$
$$n_2 \rightarrow n_4 : \text{packets with destination } \mathcal{E}_{k_n}(n_c)$$
$$n_2 : \text{keeps packets with destinations } \mathcal{E}_{k_n}(n_l),\ l \neq a, b, c$$

2) Privacy Protection Analysis: In the following, we present how B-FwHider anonymizes the forwarder information and prevents attacks introduced in Section III-B2.

Anonymizing Packet Forwarder: We see that only steps ***(c)***, ***(d)***, and ***(e)*** of B-FwHider involve the forwarder information. In step ***(c)***, the forwarder information is encrypted by the key of n_2 (i.e., k_2), which is not known by the head node. In step ***(d)***, n_2 only knows the forwarder for each k_n-encrypted destination. In step ***(e)***, packets are relayed by n_2, so neighbor set nodes cannot know actual forwarders for those packets. Consequently, after the packet forwarding in B-FwHider, each node only knows the destinations for which it is the forwarder and cannot know those of other nodes, thereby anonymizing the forwarder information.

Eavesdropping: B-FwHider imports an encryption key for the neighbor set, i.e., k_n, which is shared among neighbor set nodes without any protection. This makes B-FwHider suffer from the eavesdropping. For example, a malicious node can learn the key k_n through eavesdropping. Then, with key k_n, it can know other nodes' utilities or the forwarder information through eavesdropping again in step ***(b)***, ***(c)***, and ***(d)***.

Probing Attack and Brute-force Attack: B-FwHider can naturally resist the probing attack since the forwarder information is anonymized. The method used in E-ReHider (Section IV-B1) can also be used to further enhance B-FwHider's resistance to the probing attack.

B-FwHider can also prevent the brute-force attack.This is because in step ***(c)*** of B-FwHider, n_1 receives the group utilities instead of its own utilities from n_2. Then, since the order of the utilities values in the neighbor set utilities is different with that of n_1's utilities, pairs of cipher-text and clear-text cannot be collected for the brute-force attack.

3) Summary: B-FwHider can effectively anonymize the forwarder information. It can resist the probing attack and the brute-force attack but suffers from the eavesdropping attack. We propose E-FwHider to handle this problem in Section V-B.

B. Enhanced Forwarder Anonymity (E-FwHider)

E-FwHider is designed to thwart the eavesdropping attack in B-FwHider. As introduced in the previous subsection, k_n can be eavesdropped by the attacker. Then, it can decrypt the eavesdropped messages to derive node utilities and forwarder information.

We solve this problem by securing the communication among neighbor nodes. Specifically, in E-FwHider, nodes in the neighbor set adopts the Diffie-Hellman key exchange protocol [29] to establish a symmetric key, denoted by k_g. The Diffie-Hellman key exchange protocol enables two nodes to securely distribute a symmetric key even over an insecure communication channel (e.g., eavesdropped by others). Due to the page limit, please refer to [29] for details of this protocol. Then, all messages transmitted among neighbor nodes are further encrypted with key k_{g2}. Consequently, the group key k_n cannot be obtained by an eavesdropper. Thus, E-FwHider effectively thwarts the eavesdropping attack. Moreover, E-FwHider also prevents nodes from obtaining other's private information through collusion since each pair of nodes (i.e., $\{n_g, n_2\}$) has a unique key k_{g2}.

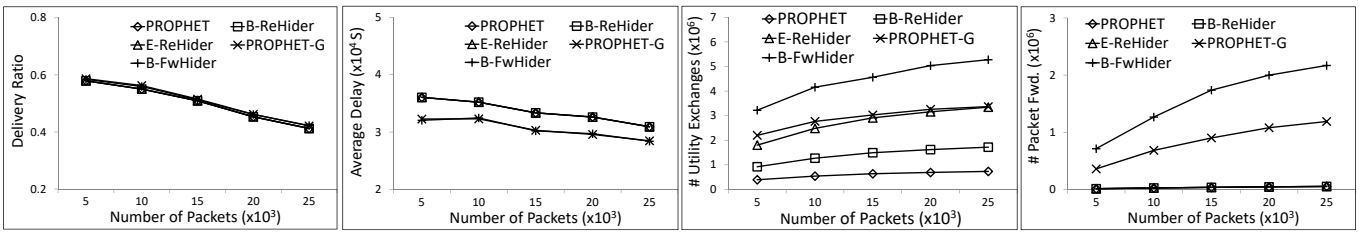
VI. Performance Evaluation

The proposed strategies' effects on privacy protection have been analyzed in Sections IV-A2, IV-B, V-A2, and V-B. We now evaluate their influences on routing efficiency and energy consumption through real-trace driven simulation and real deployment on smartphones.

A. Routing Efficiency

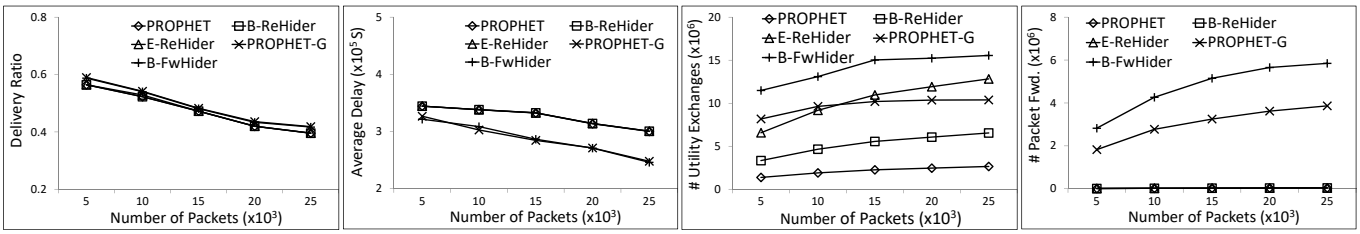
1) Experiment Settings: We used the PROPHET algorithm [2] in the test to represent the utility-based DTN routing algorithms. We conducted event-driven experiments with two real traces: Haggle trace [30] and MIT Reality trace [31]. The former records the encounters of 98 researchers at Infocom 2006 while the latter contains those of 94 students at MIT. We measured delivery ratio, average delay, the number of utility forwarding, and the number of packet forwarding in the test. The first two refer to the ratio of successfully delivered packets and their average delay. The latter two denote the total number of utility forwarding and packet forwarding.

We use the XOR operation with a key as the commutative encryption algorithm and the multiplication with a key as the hash function. Whenever a node meets another node for packet routing, it generates a new key randomly. When testing FwHider, since the traces only provide pairwise encountering records, we assume that when two nodes meet, they also meet each other's neighbors. We use PROPHET-G to denote PROPHET in the test with such an assumption. We use B-ReHider, E-ReHider, B-FwHider to denote PROPHET with corresponding strategies. We did not show the results of E-ReHider because it is the same as B-ReHider except that it has additional overhead on the secure key distribution.



(a) Delivery ratio. (b) Average delay. (c) # of utility exchange. (d) # of packet forwarding.

Fig. 3: Performance of each method with the Haggle trace with different number of packets.



(a) Delivery ratio. (b) Average delay. (c) # of utility exchange. (d) # of packet forwarding.

Fig. 4: Performance of each method with the MIT Reality trace with different number of packets.

2) Experiment Results: The test results with the two traces are shown in Figure 3 and 4. We observe from the two figures that the routing performance on delivery ratio and average delay remains the same after adopting privacy protection strategies, i.e. B-ReHider and E-ReHider have the same routing performance with PROPHET, and B-FwHider has the same routing performance with PROPHET-G. This demonstrates that the proposed strategies can correctly compare utilities and forward packets. We also see that B-FwHider and PROPHET-G have slightly higher delivery ratio and lower delay than PROPHET, B-ReHider, and E-ReHider. This is because in the tests with B-FwHider and PROPHET-G, we assume encountering nodes can also meet each other's neighbors, leading to more packet forwarding opportunities.

Figure 3(c) and 4(c) show the numbers of utility forwarding with different strategies. We find that B-ReHider roughly doubles the number of utility forwarding in PROPHET, E-ReHider doubles the number of utility forwarding in B-ReHider, and B-FwHider generates the most utility forwarding. We see that these privacy protection strategies do not significantly increase the utility exchange, which demonstrates their applicability.

Figure 3(d) and 4(d) demonstrate the numbers of packet forwarding with different strategies, which follow PROPHET≈B-ReHider≈E-Re Hider<PROPHET-G<B-FwHider. B-ReHider and E-ReHider do not change how packets are forwarded, thus having the same amount of packet forwarding with PROPHET. In PROPHET-G and B-FwHider, we assume that a node can also communicate with the newly encountering node's neighbors, which leads to much more packet forwarding opportunities. Packets in B-FwHider are relayed by the relay node to anonymize the forwarder information, leading to the most packet forwarding. However, we find that the extra packet forwarding in B-FwHider is not significantly larger than that in PROPHET-G. This shows that B-FwHider has acceptable overhead for additional protection on node privacy.

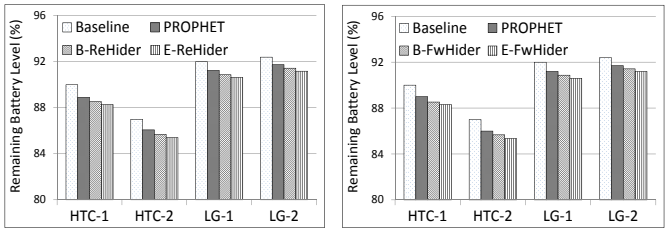
B. Energy Consumption

We further deployed ReHider and FwHider on Windows smartphones to evaluate the energy consumption in practice. We used two HTC Surround smartphones and two LG Quantum smartphones for test, denoted by HTC-1, HTC-2, LG-1, and LG-2. To simulate the interaction among encountering nodes in a DTN, each Windows phone repetitively establishes connections with virtual nodes developed in a desktop through WiFi connection to compare utilities and forward packets. The encryption algorithm and hash function were the same as those in the simulation in Section VI-A.

We assumed that there are 100 destination nodes in the network. We ran 400 rounds of interaction in each test. In each interaction, each phone connected to one virtual node in ReHider and S_f virtual nodes in FwHider. S_f was randomly selected from [1,9]. In a connection, N_p packets for N_d randomly selected destinations were generated. N_p and N_d were randomly selected from [1,40] and [1,30], respectively. We restored each phone to the factory setting and only installed the test program. We set the screen brightness to the lowest level. We charged the phone to full before each test and used the **remaining battery level** (i.e., the percent of available battery) to reflect the energy consumption in the test.

1) ReHider: In this test, we use PROPHET, B-ReHider and E-ReHider to represent original PROPHET without privacy protection, PROPHET with B-ReHider, and PROPHET with E-ReHider, respectively. We also test the Baseline scenario, in which no program is running on the phone, to reflect the background energy consumption. We ran each scenario 5 times and calculated the average remaining battery level on each phone. The results are shown in Figure 5(a).

We see from the figure that the remaining battery levels follow Baseline>PROPHET>B-ReHider>E-ReHider. Further, we see that compared to the background energy consumption



(a) ReHider. (b) FwHider.

Fig. 5: Remaining battery level after 400 rounds of interactions.

in Baseline, B-ReHider and E-ReHider only incur slightly higher energy consumption, i.e. around 5% to 10%. Such a result demonstrates the applicability of B-ReHider and E-ReHider.

2) FwHider: In this test, we use PROPHET, B-FwHider, and E-FwHider to represent original PROPHET, PROPHET with B-FwHider, and PROPHET with E-FwHider, respectively. The Baseline method is the same as in the previous test. We ran each scenario 5 times to get the average result, which is shown in Figure 5(b). We see that the remaining battery levels follow Baseline> PROPHET>B-FwHider>E-FwHider, which is aligned with that in Figure 5(a) for ReHider. B-FwHider and E-FwHider only generate slightly higher energy consumption than PROPHET (5% to 10%).

All above results demonstrate the applicability of B-FwHider and E-FwHider on current smartphones.

VII. Conclusion

In this paper, we propose two strategies, namely ReHider and FwHider, to protect the private information in routing utilities and forwarder information in utility-based DTN routing algorithms. ReHider uses commutative encryption and order-preserving hashing to anonymize routing utilities between two encountered nodes. FwHider protects both routing utilities and forwarder information when a group of nodes meet by adopting the techniques in ReHider and a novel set of utility and packet forwarding procedures. We also propose enhanced versions of the two strategies that can better thwart malicious attacks. Analytical results, extensive real-trace driven experiments, and real deployment on smartphones demonstrate that the proposed strategies can effectively protect the private information without sacrificing routing efficiency. In the future, we plan to investigate the protection of private information under more complicated attacks.

Acknowledgements

This research was supported in part by U.S. NSF grants ACI-1719397 and CNS-1733596, Microsoft Research Faculty Fellowship 8300751, and the startup fund from SIU.

References

[1] S. Jain, K. R. Fall, and R. K. Patra, "Routing in a delay tolerant network," in *Proc. of SIGCOMM*, 2004.

[2] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks." *Mobile Computing and Communications Review*, vol. 7, no. 3, 2003.

[3] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "MaxProp: Routing for vehicle-based disruption-tolerant networks," in *Proc. of INFOCOM*, 2006.

[4] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "DTN routing as a resource allocation problem." in *Proc. of SIGCOMM*, 2007.

[5] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: social-based forwarding in delay tolerant networks," in *Proc. of MobiHoc*, 2008.

[6] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," in *Proc. of MobiHoc*, 2007.

[7] J. Wu, M. Xiao, and L. Huang, "Homing spread: Community home-based multi-copy routing in mobile social networks." in *Proc. of INFOCOM*, 2013.

[8] W. Gao and G. Cao, "On exploiting transient contact patterns for data forwarding in delay tolerant networks." in *Proc. of ICNP*, 2010.

[9] X. Zhang and G. Cao, "Transient community detection and its application to data forwarding in delay tolerant networks." in *Proc. of ICNP*, 2013.

[10] R. Lu, X. Lin, and X. Shen, "Spring: A social-based privacy-preserving packet forwarding protocol for vehicular delay tolerant networks." in *Proc. of INFOCOM*, 2010.

[11] K. Chen and H. Shen, "DTN-FLOW: Inter-landmark data flow for high-throughput routing in DTNs," in *Proc. of IPDPS*, 2013.

[12] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data." *IEEE TKDE*, vol. 16, no. 9, pp. 1026–1037, 2004.

[13] E. A. Fox, Q. F. Chen, A. M. Daoud, and L. S. Heath, "Order preserving minimal perfect hash functions and information retrieval." in *Proc. of SIGIR*, 1990.

[14] K. Chen and H. Shen, "SMART: Lightweight distributed social map based routing in delay tolerant networks." in *Proc. of ICNP*, 2012.

[15] R. Lu, X. Lin, T. H. Luan, X. Liang, X. Li, L. Chen, and X. Shen, "Prefilter: An efficient privacy-preserving relay filtering scheme for delay tolerant networks." in *Prof. of INFOCOM*, 2012.

[16] G. Costantino, F. Martinelli, and P. Santi, "Privacy-preserving interest-casting in opportunistic networks." in *Proc. of WCNC*, 2012.

[17] X. Lin, R. Lu, X. Liang, and X. Shen, "Stap: A social-tier-assisted packet forwarding protocol for achieving receiver-location privacy preservation in vanets." in *Proc. of INFOCOM*, 2011.

[18] G. Vakde, R. Bibikar, Z. Le, and M. Wright, "Enpassant: anonymous routing for disruption-tolerant networks with applications in assistive environments." *Security and Communication Networks*, 2011.

[19] H. Shen and L. Zhao, "Alert: An anonymous location-based efficient routing protocol in manets," *TMC*, 2013.

[20] A. C. Yao, "Protocols for secure computations," in *Proc. of FOCS*, Washington, DC, USA, 1982.

[21] Q. Li and G. Cao, "Efficient and privacy-preserving data aggregation in mobile sensing." in *Proc. of ICNP*, 2012.

[22] X. Wang, J. Zhu, A. Pande, A. Raghuramu, P. Mohapatra, T. Abdelzaher, and R. Ganti, "Stamp: Ad hoc spatial-temporal provenance assurance for mobile users." in *Proc. of ICNP*, 2013.

[23] C. B. Lafuentea, J.-M. Seigneura, W. Moreirab, P. Mendesb, L. Maknaviciusc, A. Bogliolod, and P. Di Francescoe, "Trust and cooperation incentives for wireless user-centric environments," *e-society*, p. 312, 2012.

[24] N. Asokan, K. Kostiainen, P. Ott, and L. C, "Towards securing distuption-tolerant networks." *Nokia Research, Tech. Rep*, 2007.

[25] H. Zhu, X. Lin, R. Lu, X. Shen, D. Xing, and Z. Cao, "An opportunistic batch bundle authentication scheme for energy constrained DTNs." in *Proc. of INFOCOM*, 2010.

[26] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems." *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[27] F. Miller, *Telegraphic code to insure privacy and secrecy in the transmission of telegrams.* C.M. Cornwell, 1882.

[28] K. Chen and H. Shen, "Fine-grained encountering information collection under neighbor anonymity in mobile opportunistic social networks." in *Proc. of ICNP*, 2015.

[29] D. Gollmann, *Computer security (2. ed.).* Wiley, 2005.

[30] A. Chaintreau, P. Hui, J. Scott, R. Gass, J. Crowcroft, and C. Diot, "Impact of human mobility on opportunistic forwarding algorithms," in *Proc. of INFOCOM*, 2006.

[31] N. Eagle, A. Pentland, and D. Lazer, "Inferring social network structure using mobile phone data," *PNAS*, vol. 106, no. 36, 2009.