Ad Hoc Networks 10 (2012) 918–930

Contents lists available at SciVerse ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

ARDEN: Anonymous networking in delay tolerant networks

Cong Shi [*], Xiapu Luo [1], Patrick Traynor, Mostafa H. Ammar, Ellen W. Zegura

College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, United States

A R T I C L E I N F O

Article history:
Received 28 February 2011
Received in revised form 26 August 2011
Accepted 30 November 2011
Available online 10 December 2011

Keywords:
Delay tolerant networks
Anonymous networking
Privacy
Attribute-based encryption

A B S T R A C T

Delay Tolerant Networks (DTNs) provide a communications infrastructure for environments lacking continuous connectivity. Such networks rely on the mobility of nodes and the resulting opportunistic connections to carry messages from source to destination. Unfortunately, exchanging packets with an arbitrary intermediary node makes privacy difficult to achieve in these systems as any adversary can easily act as an intermediary and determine the sender and receiver of a message. In this paper, we present ARDEN, an anonymous communication mechanism for DTNs based on a modified onion routing architecture. Instead of selecting specific nodes through which messages must pass as is traditionally done in onion routing, ARDEN uses Attribute-Based Encryption (ABE) to specify and manage groups that may decrypt and forward messages. Through simulation, we show that this approach not only increases throughput and reduces end-to-end latency over traditional onion routing techniques, but also adds minimal overhead when compared to DTN routing protocols that do not provide anonymity guarantees. Through this, we show that ARDEN is an effective solution for anonymous communication in intermittently connected networks such as DTNs.

1. Introduction

Delay Tolerant Networks (DTNs) extend a communications infrastructure into environments lacking continuous end-to-end connectivity. Through a combination of node mobility and store-and-forward routing protocols, DTNs exploit short-lived connections between nodes to carry messages from source to destination. Because of their flexibility, such systems have been discussed as one of the next great expansions of the Internet, reaching users in diverse settings including developing nations [1,2], inter-planetary communications [3,4] and military operations [5,6].

While significant research efforts have focused on improving networking [7,8] and security issues [9,10] for DTNs, anonymous communication has received relatively little attention. However, anonymous communication in DTNs is very important, especially in the military operations where DTNs are of great value. Such mechanisms are crucial given that the knowledge of two nodes having exchanged messages may allow an adversary to recover sensitive information, including the structure of an organization or changes in that organization's behavior [11–13]. For dissidents attempting to communicate in the presence of an oppressive government or a deployed army that worries about exposing the location of its officers, such traffic analysis attacks may be as damaging as a loss of confidentiality.

A range of mature anonymous communication methods exist for Internet and MANETs. Unfortunately, none of these techniques are immediately applicable to DTNs given the often unpredictable patterns of connectivity, long communication delay and scarce communication opportunities. These challenges make it especially important to emphasize the communication performance and consider the unique network properties of DTNs in the design of anonymous communication methods.

* Corresponding author. Tel.: +1 404 518 1265.
E-mail addresses: cshi7@cc.gatech.edu (C. Shi), csxluo@comp.polyu.edu.hk (X. Luo), traynor@cc.gatech.edu (P. Traynor), ammar@cc.gatech.edu (M.H. Ammar), ewz@cc.gatech.edu (E.W. Zegura).

[1] Xiapu Luo is now a research fellow at the Hong Kong Polytechnic University. This work was done when he was a research fellow at Georgia Institute of Technology.

1570-8705/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2011.11.008

In this paper, we present ARDEN, an anonymous networking protocol adapted to the unique operating conditions of DTNs. ARDEN builds upon traditional onion routing techniques [14] by wrapping messages in multiple layers of encryption in order to hide their sources and destinations from intermediary nodes. However, whereas traditional onion routing requires that messages be encrypted using the keys of specific proxy nodes, ARDEN uses Attribute-Based Encryption (ABE) and multicast communications to allow multiple nodes to forward messages toward their intended destinations. ARDEN uses the combination of these mechanisms to take advantage of DTNs in three critical ways. First, ARDEN is able to perform relatively expensive but semantically expressive cryptographic operations without reducing throughput, because the time between communication opportunities is often long enough to allow for complex computational tasks. Second, ARDEN can better respond to unexpected changes in topology by allowing specific small groups of nodes to act as proxies. Accordingly, ARDEN is able to provide DTNs with the same guarantees offered by onion routing on the Internet while drastically reducing timing correlation attacks. Third, ARDEN does not require complete knowledge of a path between source and destination, instead uses rough knowledge of groups which is easier to obtain in DTNs.

In so doing, we make the following contributions:

- *Design a robust anonymous communication protocol for DTNs:* We develop ARDEN, an anonymous networking protocol for DTNs. ARDEN builds on a traditional onion routing architecture but incorporates Attribute-Based Encryption (ABE) to simplify management and drastically reduce latency while providing strong anonymity guarantees.
- *Measure and compare performance:* We perform a number of extensive simulations to accurately characterize ARDEN using the Haggle [15] and RollerNet [16] datasets. We also compare ARDEN against traditional onion routing approaches and against routing mechanisms without anonymity guarantees to understand its impact on latency and loss. Our experiments demonstrate that ARDEN adds very low communications overhead, decreasing delivery rates by as little as 1.1% in some cases.
- *Discuss robustness to more advanced attacks:* We also investigate ARDEN's robustness to a number of passive and active attacks. We show that breaking ARDEN requires extraordinary effort on the part of the adversary.

The remainder of the paper is organized as follows: In Section 2, we present an overview of related work; Section 3 defines models for DTNs and our adversaries, and offers design goals for our solution; Section 4 presents the design overview of ARDEN; Section 5 formally defines ARDEN and its supporting cryptographic constructions; Section 6 presents the results of our simulations and compares ARDEN to systems with and without different anonymous networking mechanisms; Section 7 offers an informal analysis of the security of our system against a range of adversaries; Section 8 provides concluding remarks.

2. Related work

Delay tolerant networks (DTNs) [17] are a class of challenged networks that experience frequent, long-duration partitioning. These networks have received significant attention from the research community because they extend communication to environments where synchronous end-to-end connectivity may be never achieved. DTNs have promising applications in rural and developing regions, interplanetary and military communication, and other adversarial environments [17,18]. A range of architectures and routing algorithms have been proposed for different types of DTNs [19,20,15,7,8,21]; however, these works primarily focus on increasing throughput and reducing delay in such systems.

Anonymous communication is well studied in the Internet. Chaum [22] designed the first modern anonymity system, Mix-Net, which hides sender and receiver identities by relaying encrypted messages through a series of proxies known as mixes. Each mix decrypts, delays and reorders messages before relaying them to the next hop. Many other anonymous communication systems, ranging from those with strong guarantees and high latency [23–25] to those with less strong anonymity but low latency [26,14,27,28], are available to Internet users wishing to protect their privacy. Of all of these systems, Tor [26] is the most widely used with hundreds of thousands users. Many of these solutions, especially those aiming to provide low latency, are susceptible to traffic analysis attacks. A range of attacks, including tagging [29,25], packet counting [30,31] and timing [32] are realistic for an attacker with less information and fewer capabilities than a ubiquitous globally passive adversary.

Some anonymity systems have also been proposed for MANETs. In response to the observation by Kong et al. [33] that ad hoc routing protocols are subject to passive attacks, ANODR [34] was proposed to conceal the identities of sources, destinations and intermediate nodes. MASK [35] attempts to achieve similar goals with a reduced computational overhead. PPCS by Choi et al. [36] further reduces computation through the use of a multi-path random forwarding scheme. With the good connectivity and long-lived paths through networks, computational cost becomes the performance bottleneck of anonymous communication in MANETs. These schemes reduce the computational cost by constructing the anonymous routing paths on-demand and using them for following anonymous communication. Therefore, they require good network connectivity and long-lived paths. Our proposed scheme directly addresses this difference.

Anonymous communication in DTNs is obtaining attention from the research community. Kate et al. [37] proposed an authentication mechanism based on a combination of the Sakai–Ohgishi–Kasahara key agreement scheme [38] and hierarchical identity-based encryption (HIBE) signature schemes [39] to support anonymous communication in DTNs. Their approach relies on a *single trustable* DTN gateway, reachable by all nodes on a regular basis, to strip data's origin before relaying. Hence this scheme is very similar to an Internet-based single proxy forwarding scheme [27]. This approach suffers from single point failure. Specifically, if a DTN gateway is compromised, attackers can deanonymize all traffic. Moreover, the DTN gateway will also become the bottleneck of performance. Janse and Beverly [40] proposed a threshold pivot scheme that uses threshold secret sharing [41] to encrypt and decrypt the anonymous messages. The receiver's address and the message can be decrypted only when more than τ of s secrets are collected. Every secret is shared among a predetermined set of nodes. Without specifying the routing path, this scheme is able to avoid those hops with long delay and, thus, reduce end-to-end latency. However, since every pair of incoming and outgoing anonymous messages on a relay node have identical portions in this scheme, attackers can easily track the messages. Furthermore, the attackers can obtain all essential secrets through sybil attack [42] and decrypt all anonymous messages. ALAR [43] was proposed to avoid revealing the sender's physical location through message fragmentation. However, it does not protect the sender ID. Accordingly, there exists a need for new and more robust schemes in the area of anonymous communication for DTNs.

3. Assumptions and design goals

3.1. DTN network model

A DTN is composed of a set of nodes. The period of time when two nodes are within direct communication range is called contact. Within a contact, nodes can transfer bundles (i.e., DTN packets) to each other. The duration and transfer bandwidth of a contact is limited. A node can deliver bundles to a destination node directly if within radio range or otherwise via intermediate nodes. Nodes have a limited buffer space to temporarily store the in-transit bundles. When a node's buffer is full, it will drop bundles. Destination nodes are assumed to have enough storage for delivered bundles. Finally, we assume that a node is able to obtain the IDs of a large proportion of nodes in the DTN, which is essential to onion routing and its variations. It can be achieved by periodically disseminating node information [44]. In fact, most DTN routing algorithms (e.g., ED [7], MaxProp [20], PROPHET [45], OPF [21], etc.) share and rely on node information for efficient bundle forwarding. Some others (e.g., RAPID [19]) even assume the existence of a control channel for metadata sharing.

Formally, a DTN is represented by a multi-graph $G = (V,E)$, where V and E are the set of nodes and edges, respectively. Each edge e represents a contact between two nodes. It can be annotated with a tuple (n_1,n_2,t,d,b), where n_1 and n_2 are the two nodes, t is the start time of the contact, d is the duration of the contact, and b is the bandwidth. A bundle is represented by a tuple (u,v,s,t), where u and v are the source node and destination node respectively, s is its size, and t is the creation time. A DTN routing algorithm delivers packets based on a feasible schedule of packet transfer, where feasible means the total number of bytes transmitted per contact is $\leqslant b \times d$. There are several different types of DTN routing algorithms with varying latency and loss properties, which we will discuss in Section 6.

3.2. Design goals

Anonymity: ARDEN is designed to provide sender and receiver anonymity. We define *strong sender anonymity* as the inability of any node other than the source, including the receiver, to determine the origin of a message with greater than negligible probability (i.e., $\frac{1}{n}$, where n is the total number of nodes in the DTN). We note that a weaker definition of sender anonymity is possible in which the destination is able to determine the source without affecting unlinkability (defined below). *Receiver anonymity* is the dual of sender anonymity and prevents any nodes except the sender and receiver from learning the receiver of a message with greater than negligible probability. When these two properties hold, ARDEN provides *unlinkability*, or the inability for an adversary to determine that messages are being exchanged between two specific nodes. Like in traditional onion routing, unlinkability in ARDEN is strongly influenced by the amount of traffic present in the network and can potentially be lost if only a very small percentage of nodes ever transmit messages.

Latency: The latency associated with DTNs is high when compared to networks with continuous connectivity end-to-end. However, this same property is often useful when trying to achieve strong guarantees of anonymity. For example, Mix-net [22] source nodes randomly choose a set of relay nodes among all nodes, leading to high latency on every relay link. Mix nodes also wait for long periods of time to collect and shuffle enough packets before sending them to the next mix node. It is the lack of such delays that makes low latency anonymity networks such as Tor susceptible to a number of attacks [32,46]. Given the inherently long times between contacts in a DTN, we strive to provide anonymous communications without adding significant additional latency.

Redundancy: Contacts are scarce resources in a DTN. Whereas some anonymous networking systems introduce dummy traffic or flood the network by transmitting messages to all participants to frustrate traffic analysis, these classes of mechanisms are not suitable for DTNs. Instead, ARDEN carefully balances the tradeoffs between redundancy, loss and anonymity through the use of multicast communications.

3.3. Threat model

We model the security of our system against a very powerful adversary. We assume that an attacker can monitor all radio communications across the range of the DTN, even though the network administrator cannot (i.e., a globally passive adversary). Moreover, an attacker may control a subset of nodes and use them to collude against legitimate nodes. Compromised nodes can set up zero-latency channels between each other and are therefore more capable than uncompromised nodes.

Nodes in our system use strong cryptographic algorithms with sufficiently long key lengths to prevent practical cryptanalysis attacks from uncovering the source, destination or contents of a message. As we explain in Section 4, properties of the Attribute-Based Cryptosystem we rely upon prevent compromised nodes from combining their keying material to gain access to messages not accessible by any individual colluder. Finally, modifications to messages can be readily detected using cryptographic hash algorithms.

4. Design overview

In this section, we present a high-level overview of the ARDEN design.

4.1. Onion routing overview

First, we provide a brief, high-level description of onion routing [14] as it forms the foundation of ARDEN. A source wishing to talk anonymously to a destination encrypts a message multiple times. As shown in Fig. 1, the message is first encrypted with the keys corresponding to Layer 1, then Layer 2 and finally Layer 3. Each "layer" of encryption uses a different key that is shared with a node on the path between source and destination. As the message is passed along the path, each hop decrypts the incoming packet with its key to remove a layer and then forwards the modified packet to the next hop. On delivery to the destination, the message is fully decrypted.

4.2. ARDEN design

The goal of ARDEN is to provide an efficient anonymous communication mechanism for DTNs. To achieve this, ARDEN builds upon a foundation of onion routing and modifies it for disconnected environments. The lack of long-lived paths between contacts makes the single path of onion routing susceptible to performance degradation in DTNs. Further, the complete knowledge of the *exact* path is hard to obtain in advance. ARDEN overcomes these issues by replacing single-node proxies with groups and through the use of multicast. Every node of a group can decrypt the corresponding layer of the wrapped bundle while only the intended destination can ultimately recover the final encrypted message. Through the communication redundancy provided by multicast, ARDEN manages to find a short path from senders to receivers. Fig. 2 illustrates this strategy at a high level. Note that for every relay hop ARDEN relies on the underlying DTN routing protocol to deliver anonymous bundles. Bundles are forwarded beyond their receivers to make receiver identification difficult for an adversary.

In summary, ARDEN makes a few modifications to onion routing. First, to improve anonymity, only the intended destination is able to remove the final layer of encryption. Second, to make identification more difficult, the intended receiver is not necessarily the final hop in the path. We discuss how we implement these mechanisms in Section 5.

Our ARDEN design intends to achieve a desirable trade-off between communication efficiency and anonymity in DTNs. Similar to all anonymous routing schemes, ARDEN increases the number of hops compared with the native DTN routing protocols. Through this, it trades off a little communication efficiency (e.g., successful delivery rate and delivery delay) for better communication anonymity. On the other hand, ARDEN can achieve better communication efficiency than traditional onion routing because its multiple anonymous routing paths between the senders and receivers increase the possibility of finding a short path. We will evaluate the system performance in Section 6.

4.3. Group partitioning and management

ARDEN relies on the presence of groups capable of decrypting and forwarding traffic to its receiver. In order for this approach to work, it is necessary to divide nodes into groups. We rely on dynamic group partitioning, as it allows senders to specify groups as they see fit and prevents an adversary from necessarily learning group membership. The use of a public key/certificate-based approach to this problem makes this approach more difficult, as a DTN with n nodes would require the creation of $2^n - 1$ groups, with nodes belonging to and therefore storing 2^{n-1} certificates.

We instead make use of a group management mechanism based on Attribute-Based Encryption (ABE) [47–49]. ABE allows users to combine a set of semantically

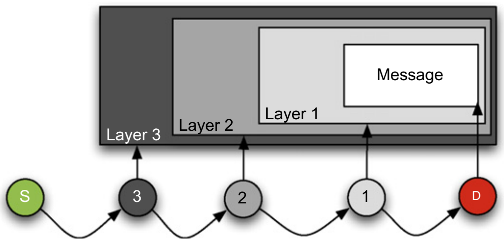


Fig. 1. A conceptual view of onion routing. A message is encrypted with multiple "layers" of encryption (here, three), each using a different key. Nodes between the source and destination can each remove a layer by decrypting the incoming message. This construction prevents a single node from learning anything other than the previous and next hops of a message.

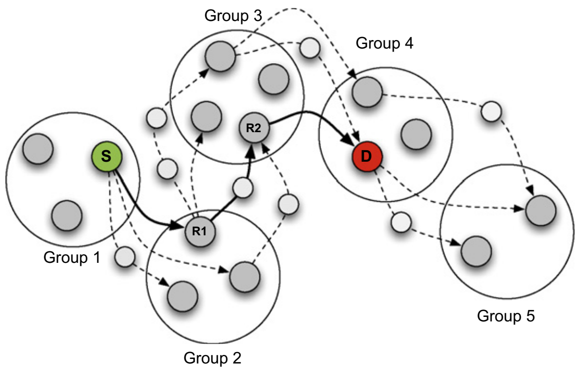


Fig. 2. An ARDEN relay path from S to D consisting of multiple forwarding paths, among which the bold path is the shortest. Note that every relay hop may consist of several intermediate nodes. The shortest anonymous relay path is not necessarily the shortest path from S to D.

expressive attributes to generate public and private keys. With ABE the administrator should choose a master secrete key (*AMK*), a public key (*APK*) and an attribute set (*P*). A user secret key (ASK_S) is generated with *AMK*, *APK* and a subset $S(\subseteq P)$. A message encrypted with *APK* and an access structure, $A \subseteq 2^P - \emptyset$, can only be decrypted by those user secret keys ASK_S that $S \in A$. In a social network, for instance, such primitives could be used to protect messages between group members (e.g., *Above 6' tall* and *Basketball Player*) and prevent users without these attributes from decrypting these messages. Moreover, these primitives prevent colluding adversaries (e.g., one *Above 6' tall* and one *Basketball Player*) from meaningfully combining their attributes and decrypting the contents of a message. As described below, we transform the binary representation of each node's ID into a set of attributes over which decryption policy can be expressed.

We specifically rely on a *Ciphertext Policy Attribute-Based Encryption* (CP-ABE) scheme for our construction [47], but refer to this approach simply as ABE throughout the paper as other related techniques could provide similar protection with minor modifications to ARDEN. When joining a DTN, a node is assigned a unique numerical identifier, a corresponding set of attributes based on the binary representation of that identifier and a user secret key based on those attributes, as shown in Fig. 3. With this scheme, knowing a node's ID allows us to define an access structure and encrypt a message that can only be decrypted by that node. We note that while more complex decryption policies are expressible in this system (e.g., *k* out of *n* thresholds, nested logical **AND** and **OR**), our structure relies strictly on the logical **AND** of multiple attributes (i.e., possession of an entire set). Under such a scheme, nodes will only need to store $\lceil log_2(n) \rceil$ attributes in order to communicate. We discuss how individual groups are specified in Section 5; however, we offer the following example policy as intuition for the reader. A sending node could specify that the first group to decrypt its bundle have IDs with a range from 16 (10000) to 23 (10111) by encrypting under the policy $P(b_0, a_1)$ (see Fig. 3). This policy ensures that only nodes with the attributes corresponding to 1 in the position 2^4 and 0 in the position 2^3 (and any other value for $2^2 - 2^0$) can decrypt the contents of the bundle.

Key revocation is a hard problem for all systems, especially DTNs. Seth et al. [50] recommend identity based encryption for DTNs because of its easy key distribution which is also achieved by ARDEN. Unfortunately, it does not solve the problem of key revocation. We will investigate efficient mechanisms as part of our future work.

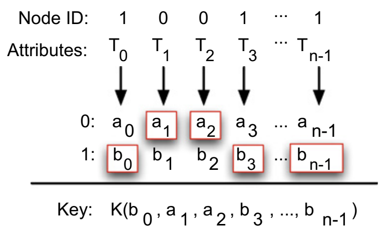


Fig. 3. The mapping between node ID, attributes and ABE user secret key.

4.4. Hop-by-hop anonymity

ARDEN uses multicast to communicate on a hop-by-hop basis. The advantage to this approach is that the next hop receiver is not necessarily identifiable to a passive adversary. Unfortunately, this approach alone is not sufficient as multiple multicast transmissions of the same bundle will appear the same to an adversary. As shown in Fig. 4, by monitoring the sender's outgoing bundles, the attackers can observe the same encrypted bundle being transmitted multiple times. To solve this problem, every node on the relay path encrypts each anonymous bundle for a single intended receiver in the next group. Accordingly, all outgoing bundles will look unique to passive adversaries. Only nodes in the next group can identify the duplication after decrypting it with their private key. The use of multiple unique-looking transmissions as a means of providing redundancy therefore also makes attacks by a passive adversary significantly more difficult.

5. Formal protocol description

In this section, we formally define the ARDEN protocol.

5.1. Notation

ARDEN employs cryptographic primitives including symmetric encryption and attribute-based encryption. We use the following notation:

- m: The message to be sent. All messages are of the same size through either padding or splitting.
- $[m]_K$: m's Keyed-Hash Message Authentication Code (HMAC) with key *K* (e.g., HMAC-SHA-384).
- `SKeyGen ()`: symmetric key generator.
- $\{m\}_K$, $\{c\}_{K^{-1}}$: Symmetric encryption of m or decryption of c with key *K* (e.g., AES).
- `APK`, `AMK`, `ASK`, $\mathbb{A}$: ABE public key, master secret key, user secret key and the corresponding access structure.
- `AEncrypt (m,APK,`$\mathbb{A}$`)`: ABE encryption of m with key `APK` and access structure $\mathbb{A}$.
- `ADecrypt (c,APK,ASK)`: ABE decryption of c with public key `APK` and secret key `ASK`.

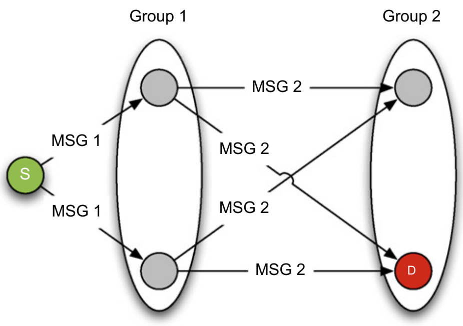


Fig. 4. An efficient traffic analysis attack at naive group-based onion routing. By associating message MSG1, attackers identify nodes in Group 1. Then by associating message MSG2, they can identify the next group, Group 2.

- G_i, S_i $(i = 1 \ldots L \ldots N)$, $\mathfrak{L} \leqslant L \leqslant N$: Members in the ith group and the number of members. There are totally N groups and the receiver is in the Lth group. For the sake of anonymity, L should not be less than a threshold $\mathfrak{L}$ whose default value is 4. While L could be equal to N, a better receiver anonymity could be achieved if $L < N$.
- $\mathbb{D} = \{d_1, \ldots, d_{\mathfrak{N}}\}$: Node ID and $d_i \in \{0,1\}$.
- $\mathbb{T}_{\mathbb{D}} = \{t_1, \ldots, t_{\mathfrak{N}}\}$: All attributes of a node.

$$t_i = \begin{cases} a_i, \text{if } d_i = 0 \\ b_i, \text{if } d_i = 1 \end{cases} \tag{1}$$

a_i and b_i $(i = 1 \ldots \mathfrak{N})$ are attributes.

- θ: Padding in a bundle.

5.2. Network initialization

Before setting up a DTN network, the administrator generates an ABE public key (APK) and master secret key (AMK). APK is given to all nodes while the administrator keeps the master secret key private. When a new node joins the DTN, it is randomly assigned a unique ID $\mathbb{D}$, and the corresponding user secrete key (ASK) as shown in Fig. 3. The node obtains the IDs of other nodes in the DTN from the administrator and other nodes through a gossip protocol [44].

5.3. Defining groups

Algorithm 1. Dynamic group partitioning

```
1: procedure PARTITIONGROUPS($S,k$) ▷ $S$ is the set of nodes;
   $k$ is the required group size.
2:     $seed$← Random ();
3:     $tree$← InitTree ();
4:     for $node$ in $S$ do
5:         $node.groupingKey$← shuffle ($node.\mathbb{D}, seed$);
6:         $tree.Insert(node)$; ▷ Insert $node$ into $tree$
   according to its $groupingKey$
7:     end for
8:     return PartitionTree ($tree,k$);
9: end procedure
10: procedure PARTITIONTREE$tree$, $k$
11:     if $tree.leafSize() < k$ then
12:         groupSet ←$null$;
13:     else if $tree.leafSize() == k$ then
14:         groupSet.add ($tree$);
15:     else
16:         leftSet ← PartitionTree ($tree.left,k$);
17:         rightSet ← PartitionTree ($tree.right,k$);
18:         if leftSet == $null$ && rightSet == $null$ then
19:             groupSet.add ($tree$);
20:         else
21:             groupSet.addAll (leftSet);
22:             groupSet.addAll (rightSet);
23:         end if
24:     end if
25:     return groupSet;
26: end procedure
```

Before sending a message through ARDEN, the sender creates N groups denoted as $G_1, \ldots, G_N$. This procedure can adopt arbitrary strategies. In our current design, ARDEN partitions groups based on the prefixes of shuffled node IDs. That is, the binary node IDs are shuffled with a random seed to generate the grouping keys (i.e., every bit in a grouping key corresponds to a bit in its node ID). All nodes are leaves of a binary tree whose positions are determined by their grouping keys. We partition the groups through depth-first searching the tree to find all disjoint subtrees that have specified group size (i.e., k). When a subtree has more than k leaves while both of its children has less than k leaves, this subtree is also partitioned as a group. Thus, the largest group has at most $2k - 2$ nodes. Algorithm 1 shows the group partitioning procedure.

Fig. 5 explains the group partitioning procedure with an example. There are 8 nodes with 4-bit IDs. We want to partition them into groups whose size is 3. We insert these nodes into a tree according to their grouping keys (i.e., shuffled IDs). Then, we search all disjoint subtrees satisfying group size requirement. Note that some nodes (e.g., $n0$ in the example) may not belong to any group.

All nodes in the same group have the same j initial bits (e.g., $\{d_0, \ldots, d_{j-1}\}$). For each group G_i $(i = 1, \ldots, N)$, the user will generate the ABE access structure $\mathbb{A}_{G_i}$ according to the common attributes of all members in that group (e.g., $\mathbb{A}_{G_i} = t_0 \wedge \cdots \wedge t_{j-1}$). $\mathbb{A}_{G_i}$ will be used to encrypt messages to the intermediary group. The sender will randomly choose several groups and the group with the receiver to create the relay path. If the receiver is not in any group, the sender will repeat the group partitioning procedure.

In current group partitioning and path selection procedures we do not consider connectivity between two sequent groups. Although it may lead to longer delay, it guarantees anonymity as adversaries cannot narrow down the possible paths using connectivity information. Moreover, increasing group size will increase the probability of attaining good connectivity between groups. It's also worth noting that a sender knowing only a small portion of node IDs still can create a relay path, although the diversity of all possible relay path is limited.

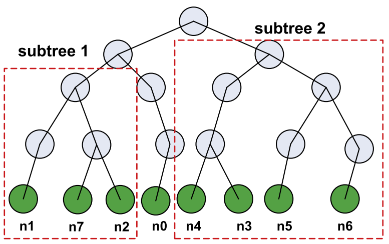


Fig. 5. An example of group partitioning. The required group size is set to 3. There are eight nodes ($n0 - n7$) with 4-bit IDs. They are inserted into the tree according to their grouping keys. Subtree 1 has exactly three nodes and, thus, is partitioned as a group. Subtree 2 has more than three nodes and should check its children. Since both of its children have less than three nodes, it is also partitioned as a group.

5.4. Constructing forward bundles

Algorithm 2 defines a forward bundle. The output of this algorithm is a forward bundle that consists of the following four major components:

Algorithm 2. Construct a forward bundle

```
1: procedure ENCRYPTBUNDLE($m, L, N, \mathbb{A}, \mathbb{A}_r$, APK)
2:     Ms., Ks, Hs, Gs ← null; $K'$← SKeyGen ()
3:     $m' \leftarrow (\{m\}_{K'}, [m]_{K'}$, AEncrypt($K'$, APK, $\mathbb{A}_r$))
4:     TKs[] ← null; FMs[] ← null
5:     for $i \leftarrow N: -1:1$ do                ▷ Encrypt symmetric keys & groups
6:         $K$← SKeyGen (); TKs[i] ←$K$; $K_G$← SKeyGen ()
7:         if $i == N$ then
8:             Gs ← $\{\theta, [\theta]_{K_G}, \theta\}_{K_G}$
9:             v ← $(K, [K]_K, K_G, [K_G]_{K_G}, \theta)$
10:        else
11:            Gs ← $\{G_{i+1}, [G_{i+1}]_{K_G}, \mathrm{Gs}\}_{K_G}$
12:            v ← $(K, [K]_K, K_G, [K_G]_{K_G}, K_s)$
13:        end if
14:        Ks ← AEncrypt (v,APK,$\mathbb{A}_{G_i}$)
15:    end for
16:    for $i \leftarrow L+1:1:N$ do                ▷ Create fake messages
17:        $K$← TKs[i]
18:        if $i == L+1$ then
19:            FMs[i] ← $\{m'\}_{K^{-1}}$
20:        else
21:            FMs[i] ←$\{FMs[i-1]\}_K^{-1}$
22:        end if
23:    end for
24:    for $i \leftarrow N: -1:1$ do                ▷ Encrypt messages and their MACs
25:        $K$← TKs[i]
26:        if $i <= L$ then
27:            if Ms. == null then
28:                Ms. ←$\{m'\}_K$
29:            else
30:                Ms. ← $\{\mathrm{Ms}\}_K$
31:            end if
32:            Hs ←$\{[\mathrm{Ms}]_K, Hs\}_K$
33:        else
34:            if $i == N$ then
35:            Hs ←$\{[\mathrm{FMs[i]}]_K\}_K$
36:        else
37:            Hs ←$\{[\mathrm{FMs[i]}]_K, Hs\}_K$
38:        end if
39:      end if
40:    end for
41:    Bundle ← (Gs,Ms,Hs,Ks)
42:    return Bundle
43: end procedure
```

- G_s: a data structure containing forwarding information, encrypted in multiple layers. Members of the *i*th group are only able to learn the identity of the $i + 1$th hop.
- M_s: the layered encryption of the message. All data is encrypted with a symmetric key, which is revealed by decrypting the corresponding level of K_s. The data itself can only be accessed by the intended destination as it is the only node with the requisite private key to uncover the symmetric key from K_s. This approach modifies the appearance of the message at each ARDEN-based hop.
- H_s: HMACs of all message layers.
- K_s: symmetric keys for encrypting G_s, M_s, H_s in different groups. The key for the ith group, its HMAC and the K_s for the $i + 1$th group are encrypted as a whole with the *i*th group's ABE public key.

The sender first encrypts the message using a symmetric key (K'), computes the message's HMAC, and encrypts the symmetric key using ABE with the receiver's access structure (line 3). Because ABE requires greater computation time than symmetric encryption, we employ ABE only to protect symmetric keys and use symmetric keys to encrypt everything else. As shown in line 5–15, we generate $2 * N$ symmetric keys. N keys are used to generate K_s and will also be used to generate M_s and H_s. The other N keys are used to build G_s.

Each component has its own HMAC to protect integrity. However, we must also generate HMAC values for groups past the intended receiver (L) to the end of the path (N) in order to allow the intended receiver to be placed randomly in the layers of the total path. As shown in lines 16–23, we use symmetric key included for nodes $L + 1$ to N to decrypt the encrypted message. In lines 24–40, we compute HMACs for both legitimate and fake messages. By doing so, nodes in the groups from $L + 1$ to N will determine that M_s has the correct HMAC value and pass it to nodes in the next group. We note that those symmetric keys used to make fake messages cannot decrypt the message intended for L but instead result in pseudorandom-looking characters.

An anonymous bundle causes a little additional overhead by introducing forwarding path, HMACs and symmetric keys (i.e., 32, 384, and 128 bits for every hop, respectively). Since bundle size is usually large in DTNs (e.g., 10 KB in many deployed DTNs [20,51]), the extra overhead (e.g., 272 B when $N = 4$) is negligible.

5.5. Creating reply paths

To prevent the receiver from identifying the sender, the sender creates an anonymous reply path G_s'' for the receiver. Similar to building G_s in Algorithm 2, group information is put in layered encryption. For example, the *i*th group's information is encrypted along with G_s'' for the $i + 1$th group using a symmetric key for the *i*th group. The sender will also include symmetric keys for the receiver to encrypt her response. The reply path information is part of the original message *m* sent from the sender to the receiver. Since both forward path and reply path are randomly constructed, specifying reply path by the sender will not hurt its routing performance.

5.6. Processing forward bundles

Intermediary nodes process incoming bundles as follows. Upon receiving a message, a hook placed in the underlying DTN routing algorithm determines whether or not the message must be processed by the ARDEN module. Non-anonymous traffic simply continues through the routing process and onto an egress buffer. The K_s field of the bundle passed to the ARDEN module is then decrypted using the node's group attributes, revealing the two symmetric keys, K_{G_s} and $K_{M_sH_s}$. The first symmetric key, K_{G_s}, is used to decrypt G_s and reveals the next hop (i.e., group) to receive the message. $K_{M_sH_s}$ removes the top-most layer of encryption of M_s. The integrity of M'_s is then checked by decrypting H_s with the same key, revealing an HMAC of M'_s and H'_s. ARDEN checks whether the HMAC of M'_s matches previously seen bundles and drops the received bundle if a collision is found. Note that this process allows intermediary nodes to legitimately drop traffic and therefore prevents an adversary from successfully executing replay attacks. Finally, before forwarding the message, the node attempts to decrypt M'_s with its private ABE key to determine if it is the intended receiver. If it is the intended receiver, the node recovers the message and then, just like other nodes in the same group it sends a new bundle to the remainder of the path (which contains G'_s, M'_s, H'_s, and K'_s).

ARDEN-processed bundles are then sent to the same egress buffer as non-anonymous traffic. Bundles stored in the egress buffer are then shuffled to remove ordering, preventing a passive adversary from strongly correlating pre- and post-processed bundles. Finally, when contact is made with another node, bundles are forwarded onto their next hop. Fig. 6 provides a high-level description of this process.

5.7. Constructing/processing reply bundles

A receiver uses the decrypted contents of M_s to construct a reply bundle. Specifically, included in M_s is a G''_s

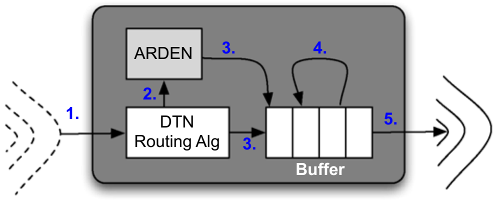


Fig. 6. The processing of incoming messages by intermediary nodes in ARDEN. After receiving a bundle (1), the underlying routing algorithm determines whether or not the bundle needs to be processed by the ARDEN module. If so (2), the bundle is decrypted and passed to the egress buffer (3). Bundles in the buffer are shuffled (4) to remove the temporal ordering and are forwarded to the appropriate next hop (5) when a new contact is made.

corresponding to the return path and a number of symmetric encryption keys. The receiver encrypts their response using the provided symmetric keys, generating the necessary HMACs for each layer, and then transmits the new bundle to the first group identified in G''_s. As each group receives the incoming bundle, they simply processes the message as before, resulting in the bundle being delivered to the original source.

Note that in order to limit the exposure of distance information, every response bundle should be encrypted in the number of keys required to traverse the entire DTN. When the original source receives an HMAC in G''_s corresponding to the original message, it simply applies the remaining keys to recover the message. In so doing, a receiver can communicate with a sender without needing to know its identity.

6. Evaluation

To better understand its characteristics and tradeoffs, we evaluate ARDEN using a range of parameters and baselines.

6.1. Experimental setup

We simulate DTNs implementing ARDEN using the Haggle [15] and RollerNet [16] datasets. Specifically, we extend DTNSim2 [51,7] with the ARDEN module. DTNSim2 is chosen because its protocol stack architecture makes it easy to implement ARDEN without modifying other protocols. However, it's not hard to implement ARDEN in other DTN simulators like the ONE simulator [52]. The Haggle dataset records the traces of 98 nodes over 94 h. We divide this trace into nine segments each lasting approximately ten hours. We remove two segments with a significantly lower number of contacts from our experiments. We conduct three experiments using each segment, each with different seed values. Our results represent the average values of the 21 experiments. The RollerNet dataset contains traces of 62 nodes in 2 sessions of 80 min. Accordingly, we divide it into two segments and executed every experiments ten times with different seeds.

Our experiments contain both anonymous and normal bundles used by various applications with different requirements on anonymity and performance. We vary both the proportion and intensity of such traffic in our experiments. All nodes generate anonymous and normal bundles destined to other randomly selected nodes, x percent of which are anonymous bundles. Every normal bundle is 10 KB and every node has a 10 MB buffer. The bandwidth of contacts is 1 MBps. Messages are not given a TTL, so loss occurs due to full node buffers. Except where differently stated, each node generates ten bundles/hr. We vary x between 0 and 20, as the number of anonymous bundles usually is smaller than that of normal bundles.

ARDEN relies on the underlying DTN routing mechanisms to deliver bundles between groups. In the experiments, we use two basic DTN routing mechanisms (i.e., earliest delivery (ED) [7] and epidemic minimum estimated expected delay (EMEED) [51] which are already implemented in DTNSim2) belonging to two popular classes of DTN routing – single path routing and replicative routing. These two algorithms demonstrate the essence of those two classes of routing mechanisms and are sufficient for our evaluation.

Traditional onion routing and the underlying DTN routing mechanisms are used as the baselines. To make the comparison between ARDEN and onion routing fair, receivers are put in the last group (i.e., layer of encryption) in all experiments. The percentage of bundles delivered and the delivery delay are two most important metrics commonly used in DTN [20,7,8], and so we record these metrics as our primary means of comparison throughout the remainder of the paper.

6.2. Experimental results

6.2.1. Dataset comparison

In the first set of experiments, we evaluate ARDEN's performance in both the Haggle and RollerNet datasets. ED is used as the underlying DTN routing mechanism. Every node in RollerNet randomly generates 100 bundles in the 80-min simulation duration so that it generates the same number of bundles as nodes in Haggle. The length of anonymous networking path is set to four and group size is three. Fig. 7 shows our results.

We make following observations. First, ARDEN achieves a much higher delivery rate when compared against onion routing in the Haggle dataset. Specifically, ARDEN delivers 17.2–24.5% more messages than traditional onion routing, while providing only a relatively small decrease in delivery rate over ED algorithm (4.7–11.6%). The delivery rate of ARDEN virtually matches the unmodified ED algorithm when using the RollerNet dataset. Specifically, ARDEN and ED deliver 95.8% and 96.9% (a decrease of only 1.1%) of their messages, respectively, whereas onion routing delivers 89.5% of anonymous messages. ARDEN's better performance on RollerNet is due this data set's higher node density.

ARDEN delivers messages with significantly lower latency than traditional onion routing in both datasets. Whereas onion routing requires an average of 131.1 min and 229.6 s in the Haggle and RollerNet datasets, respective, messages sent using ARDEN are delivered in an average of 83.9 min and 128.7 s. When compared to the ED routing algorithm, ARDEN causes an increase of only 24.5% and 39.1% for latency, as opposed to the 94.5% and 148.2% increases for traditional onion routing. The reason for this delay is due to path specification. Specifically, because the ED algorithm is able to help determine the shortest path, ARDEN may take a slightly sub-optimal path based on the groups selected at the time of message encryption. By requiring that messages pass through specific nodes, onion routing virtually guarantees that a sub-optimal path will be taken.

6.2.2. Routing algorithm comparison

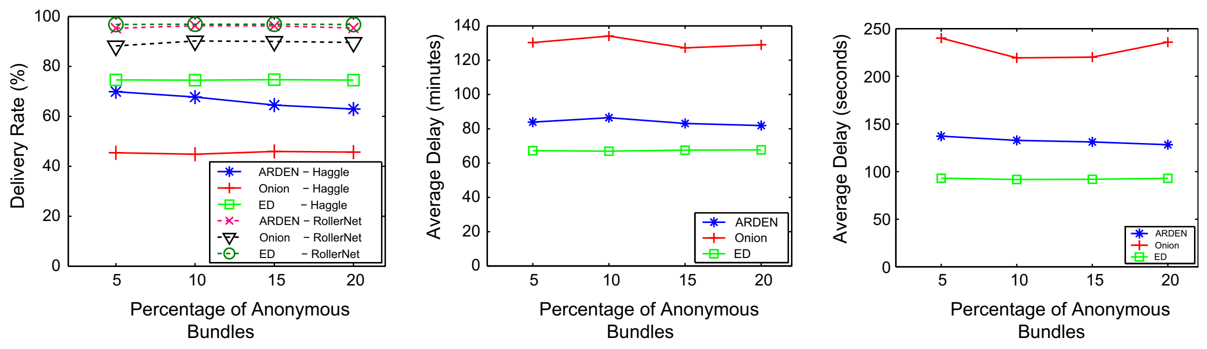
The previous experiments demonstrate that ARDEN performs relatively well when run on top of the ED routing algorithm. In this set of experiments, we use EMEED as DTN routing mechanism to analyze whether ARDEN is compatible with replicative routing. The parameter setting is the same as above experiments. Fig. 8 highlights our results.

As expected, ARDEN achieves better performance than onion routing in all the experiments using EMEED. Somewhat surprisingly, the delay in delivery for both ARDEN and traditional onion routing systems declines by 29.4% and 17.7% respectively with an increase in the proportion of anonymous bundles. The low delivery rate and declined delay indicates that the epidemic routing protocol and its replicative messaging approach helps to quickly deliver bundles with short path but hinder the delivery of bundles with long path. Through this set of experiments we demonstrates ARDEN's advantage and compatibility for various DTN routing mechanisms.

6.2.3. Traffic intensity

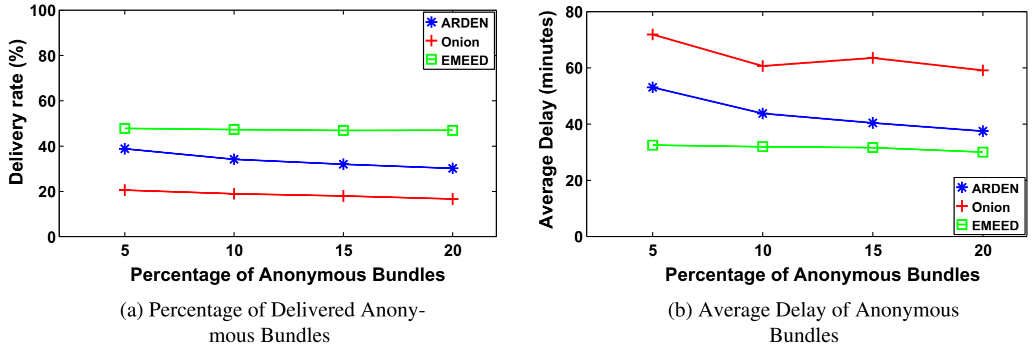
The previous experiments demonstrate that ARDEN performs relatively well in the presence of moderate traffic levels. To determine whether any inherent characteristics of the protocol harm its performance as traffic intensity increases (e.g., the use of paths extending beyond the intended receiver), we perform experiments with a message generation rate of 30 bundles/hour (i.e., three times the previous experiments) on the Haggle dataset. Fig. 9 shows the results of these simulations.

The delivery rate falls for unmodified ED (5.0%), onion routing (6.3%) and ARDEN (11.9%) when compared to Fig. 7a. This results is expected as both overfull buffers



(a) Delivery rate in Haggle and Roller Net (b) Average delay in Haggle (c)Average delay in RollerNet

Fig. 7. ARDEN's performance for the Haggle and RollerNet datasets using ED as the underlying routing algorithm.



(a) Percentage of Delivered Anonymous Bundles (b) Average Delay of Anonymous Bundles

Fig. 8. ARDEN's performance in the Haggle dataset using EMDEED as the underlying routing algorithm.

and the likelihood of wireless collisions are increased under these circumstances. Interesting, the latency for messages successfully delivered end-to-end is virtually identical to the previous experiments. While seemingly counterintuitive, this result is a product of the characteristics of DTNs. Given sufficient contact bandwidth, messages buffered in nodes are not being forced to wait across multiple contacts to be forwarded to the next hop. Accordingly, the most significant factor influencing latency for successfully delivered packets remains the topology itself.

6.2.4. Path length and group size

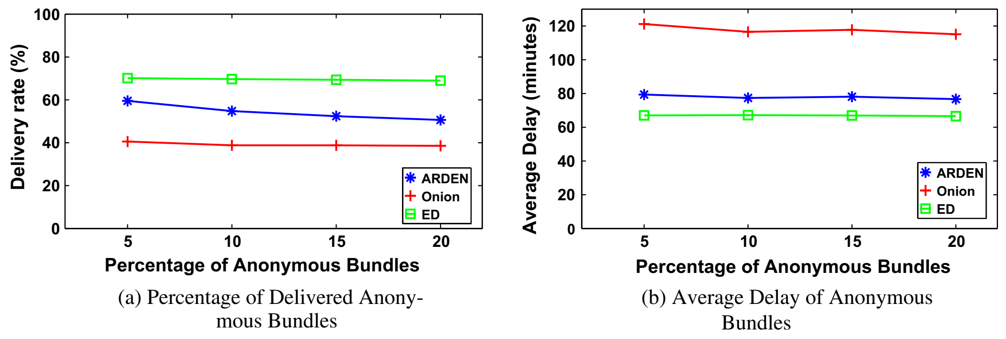
Next, we explore the performance of ARDEN when varying path lengths and group sizes are used. These parameters are arguably the most important factors for ARDEN as they can create tradeoffs between anonymity and efficiency (e.g., larger groups are more efficient, but increase the probability of a compromised node being able to decrypt a message). All experiments in this section are conducted on Haggle dataset.

Fig. 10 shows the impact of path length on delivery and latency given a moderate traffic intensity. As expected, an increase in the number of hops decreases the likelihood with which a bundle will be delivered to its intended destination. However, this decrease is not significant, dropping from approximately 68% to 62% from three to seven hops. Note that even at seven hops, ARDEN delivers a higher percentage of bundles than onion routing at four hops. Also as expected, latency increases as the number of hops grows, with an average increase of approximately three minutes per hop in our experiments. We note that because of the variety of topologies tested, the initial two hops often take the longest amount of time; however, we expect this observed early latency to be more evenly present in datasets with more uniform distributions.

Fig. 11 compares the number of nodes in a group to delivery rate and delay in the Haggle dataset with a fixed path length of four hops. With group sizes of one, ARDEN degenerates to traditional onion routing and exhibits the same delivery and delay characteristics. These values improve as group size increase, but only to a point. After groups contain three members, the decrease in latency and increase in delivery rate plateau and their differences become statistically indistinguishable. While a node will increase the probability of an adversary being part of a group by increasing the size of that group, they will not improve the delivery characteristics of their message by growing the group above three nodes. While this exact value will vary between datasets, it shows that small ARDEN groups can significantly improve delivery rate and latency compared to traditional onion routing.

6.2.5. Computational cost of ARDEN

As a final set of experiments, we evaluate the computational cost of ABE and its impact on the overall performance of ARDEN. We adopt the ABE toolkit designed and implemented in [47]. All the experiments are carried out on a 2.53 GHz Lenovo ThinkPad T400 laptop with 2 GB memory running Ubuntu 10.10. In a DTN with n nodes, every node has $\lceil log_2(n) \rceil$ attributes for ABE. We change



(a) Percentage of Delivered Anonymous Bundles (b) Average Delay of Anonymous Bundles

Fig. 9. ARDEN's performance with heavy traffic (30 bundles/hr) in the Haggle dataset using ED as the underlying routing algorithm.

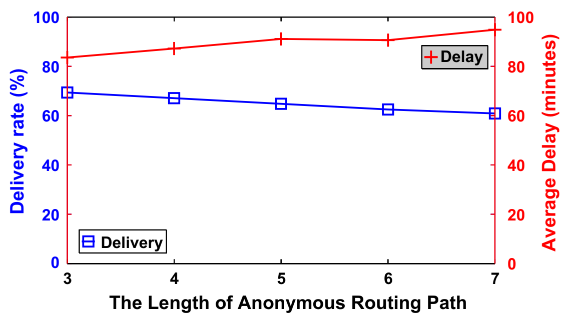


Fig. 10. The performance of ARDEN with various path lengths. ED is used for DTN routing. Haggle dataset is used for simulation.

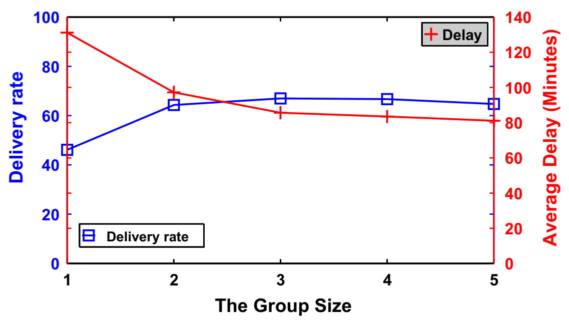


Fig. 11. The performance of ARDEN with various group size. ED is used for DTN routing. Haggle dataset is used for simulation.

the node number from 2^4 to 2^{20} and plot the average execution time of ABE operations in Fig. 12.

In a small DTN with 16 nodes, it takes ARDEN 65 ms and 44 ms for every ABE encryption and decryption operations, respectively. Meanwhile, an extremely large DTN with 1 million nodes (i.e., 2^{20} nodes) averages 393 ms and 214 ms for ABE encryption and decryption. In a typical setting where the path length is 4, every anonymous message needs 4 ABE encryption and 4 ABE decryption. Thus, the total computational cost of an anonymous message is 436 ms and 2428 ms for DTNs with 16 nodes and 1 million nodes, respectively. Compared with the long communication delay in DTNs, the computational cost of ABE is very small and has limited impact on ARDEN. For instance, in both Haggle and RollerNet datasets the average computational cost of an anonymous message is 924 ms, while their delivery delays are 83.9 min and 128.7 s, respectively, as shown in Fig. 7. The computational time only accounts for 0.018% and 0.7% of the total delay, respectively.

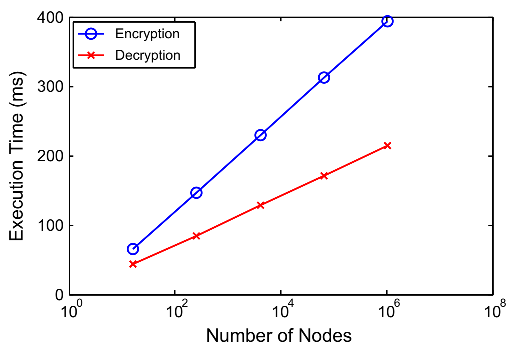


Fig. 12. The execution time of ABE operations in ARDEN. Even when there are 1 million nodes in the DTN, it takes only 393 ms and 214 ms to encrypt and decrypt a message, respectively.

7. Security analysis

7.1. Passive adversaries

ARDEN achieves sender anonymity in the presence of a passive adversary through three mechanisms. First, as messages are encrypted multiple times, an adversary cannot extract the original message and link it to the sender. Second, because all nodes can act as sources, destinations and intermediary proxies, transmitting nodes are equally as likely to be forwarding messages as they are generating them. Finally, the use of anonymous reply paths makes it difficult for a receiver to identify the sender (unless the sender explicitly identifies itself in the message).

Scenarios in which nodes rarely (if ever) transmit improve the odds of a globally passive adversary correctly guessing the source of a message. Our current implementation of ARDEN does not account for this issue; however, it can easily be addressed through the occasional generation of dummy traffic. While we previously noted that the use of dummy traffic is not appropriate in DTNs in the presence of high volumes of legitimate traffic, the creation of dummy packets when ingress throughput drops below a threshold may help to prevent such attacks. For instance, if a node receives only one bundle per quantum, it can generate a garbage message destined for a random receiver. Such *reactive cover traffic*, which can be generated at different volumes, makes traffic analysis attacks extremely difficult without wasting contact bandwidth.

ARDEN also achieves receiver anonymity through three methods. Layered encryption allows only the ultimate receiver to determine the target of a message. Because the last layer of a message is encrypted using the private key of the destination, only the destination will know that it is the intended receiver. Moreover, multicasting prevents an adversary from determining the intended receiver on a hop-by-hop basis when multiple nodes are within transmission range. Finally, by placing the receiver randomly in the layers of encryption as opposed to the deepest layer as is traditionally done in onion routing, it becomes much more difficult for an adversary to determine when the true destination has actually received a message.

7.2. Active adversaries

While robust against passive adversaries, it is important to understand how ARDEN withstands more active attacks. We therefore briefly explore a number of different attacks commonly discussed in this space.

As is the case for benign nodes, an adversary controlling a single randomly placed node will be unable to extract the source and destination of a message because of the use of multiple layers of encryption. An adversary able to compromise multiple nodes may not necessarily be more successful. For instance, compromising the nodes surrounding an arbitrary node will not reveal the source or destination of a transmitted message. An adversary will instead need to control nodes within every group along the path between a suspected source and destination in order to identify the source correctly – a physically far more demanding task. Note that because the destination node will continue to forward the bundle (potentially over multiple hops), such an attack is unlikely to successfully identify the destination of a message.

An active adversary may also attempt to manipulate the bundles exchanged between legitimate nodes. For instance, an adversary may replay specific bundles to determine when that message is retransmitted by the intermediary node. By checking to see if an incoming message matches the stored HMAC of previous messages, such attacks are easily defeated by ARDEN. An adversary attempting to watermark specific flows through the use of temporal perturbation attacks [32,46] will also fail due to both the shuffling of message within ARDEN intermediary nodes and the store-and-forward nature of DTNs. Deducing the origin of a message through dropping attacks is similarly difficult given both the long retransmission timeouts and that such retransmissions will appear different from the original message.

8. Conclusion

In this paper, we proposed ARDEN, an anonymous networking protocol tailored for use in DTNs. ARDEN relies on a modified foundation of onion routing, multicast communication and Attribute-Based Encryption (ABE) to provide source and destination anonymity. Through extensive simulation using both the Haggle and RollerNet datasets, we demonstrate that this approach not only vastly outperforms traditional onion routing in this setting, but also adds low additional overhead when compared to traditional routing mechanisms. Finally, we demonstrate the robustness of our approach to a range of adversaries both passive and active. In so doing, we show that ARDEN is an effective means of achieving anonymous communication in a DTN.

Acknowledgments

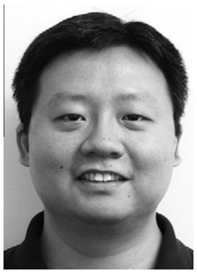
This work was supported in part by the US National Science Foundation (CCF-0916031 and CNS-0831714). Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] A.S. Pentland, R. Fletcher, A. Hasson, DakNet: rethinking connectivity in developing nations, IEEE Comput, 2004.
[2] S. Guo, M.H. Falaki, E.A. Oliver, S. Ur Rahman, A. Seth, M.A. Zaharia, S. Keshav, Very low-cost internet access using kiosknet, SIGCOMM Comput. Commun. Rev., 2007
[3] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, H. Weiss, Delay-tolerant networking: an approach to interplanetary internet, IEEE Commun. Mag., 2003.
[4] W. Ivancic, L. Wood, P. Holliday, W. Eddy, D. Stewart, C. Jackson, J. Northam, Experience with delay-tolerant networking from orbit, in: Advanced Satellite Mobile Systems, 2008 (ASMS 2008), 2008.
[5] P. Marshall, DARPA progress towards affordable, dense, and content focused tactical edge networks, in: IEEE MILCOM, 2008.
[6] C. Rigano, K. Scott, J. Bush, R. Edell, S. Parikh, R. Wade, B. Adamson, Mitigating naval network instabilities with disruption toler, in: IEEE MILCOM, 2008.
[7] S. Jain, K. Fall, R. Patra, Routing in a delay tolerant network, in: ACM SIGCOMM, 2004.
[8] W. Zhao, M. Ammar, E. Zegura, A message ferrying approach for data delivery in sparse mobile ad hoc networks, in: ACM MobiHoc, 2004.
[9] F. Li, J. Wu, A. Srinivasan, Thwarting blackhole attacks in disruption-tolerant networks using encounter tickets, in: IEEE Infocom, 2009.
[10] J. Burgess, G.D. Bissias, M.D. Corner, B.N. Levine, Surviving attacks on disruption-tolerant networks without authentication, in: ACM MobiHoc, 2007.
[11] C. Cortes, D. Pregibon, C. Volinsky, Communities of interest, in: International Symposium of Intelligent Data Analysis, 2001.
[12] P. McDaniel, S. Sen, O. Spatscheck, J. Van der Merwe, B. Aiello, C. Kalmanek, Enterprise security: a community of interest based approach, in: NDSS, 2006.
[13] L. Johansen, M. Rowell, K. Butler, P. McDaniel, Email communities of interest, in: Conference on Email and Anti-Spam (CEAS), 2007.
[14] D. Goldschlag, M. Reed, P. Syverson, Onion routing, Commun. ACM 42 (2), 1999.
[15] P. Hui, J. Scott, J. Crowcroft, C. Diot, Haggle: a networking architecture designed around mobile users, in: WONS, 2006.
[16] P.U. Tournoux, J. Leguay, F. Benbadis, V. Conan, M.D. de Amorim, J. Whitbeck, The accordion phenomenon: analysis, characterization, and impact on dtn routing, in: IEEE INFOCOM, 2009.
[17] K. Fall, A delay-tolerant network architecture for challenged Internets, in: ACM SIGCOMM, 2003.
[18] A. Lindgren, P. Hui, The quest for a killer app for opportunistic and delay tolerant networks, in: CHANTS, 2009.
[19] A. Balasubramanian, B. Levine, A. Venkataramani, Dtn routing as a resource allocation problem, in: ACM SIGCOMM, 2007.
[20] J. Burgess, B. Gallagher, D. Jensen, B.N. Levine, Maxprop: routing for vehicle-based disruption-tolerant networks, in: IEEE INFOCOM, 2006.
[21] C. Liu, J. Wu, An optimal probabilistic forwarding protocol in delay tolerant networks, in: ACM MobiHoc, 2009.
[22] D. Chaum, Untraceable electronic mail, return addresses and digital pseudonyms, in: Communication of ACM, 1981.
[23] C. Gulcu, G. Tsudik, Mixing email with babel, in: NDSS, 1996.
[24] U. Moller, L. Cottrell, P. Palfrader, L. Sassaman, Mixmaster protocol, July 2003, <www.abditum.com/mixmaster-spec.txt>.
[25] G. Danezis, R. Dingledine, N. Mathewson, Mixminion: Design of a type iii anonymous remailer protocol, in: IEEE Symposium on Security and Privacy, 2003.
[26] R. Dingledine, N. Mathewson, P. Syverson, Tor: the second-generation onion router, in: USENIX Security Symposium, 2004.
[27] The anonymizer, <http://anonymizer.com>.
[28] L. Zhuang, F. Zhou, B.Y. Zhao, A. Rowstron, Cashmere: resilient anonymous routing, in: USENIX NSDI, 2005.
[29] R. Pries, W. Yu, X. Fu, W. Zhao, A new replay attack against anonymous communication networks, in: IEEE ICC, 2008.
[30] A. Serjantov, P. Sewell, Passive attack analysis for connection-based anonymity systems, in: ESORICS, 2003.
[31] Z. Ling, J. Luo, W. Yu, X. Fu, D. Xuan, W. Jia, A new cell counter based attack against tor, in: ACM CCS, 2009.
[32] B.N. Levine, M.K. Reiter, C. Wang, M.K. Wright, Timing attacks in low-latency mix-based systems, in: Financial Cryptography, 2004.
[33] J. Kong, X. Hong, M. Gerla, A new set of passive routing attacks in mobile ad hoc networks, in: Milcom, 2003.
[34] J. Kong, X. Hong, Anodr: anonymous on demand routing with untraceable routes for mobile ad hoc networks, in: ACM MobiHoc, 2003.
[35] Y. Zhang, W. Liu, W. Lou, Anonymous communications in mobile ad hoc networks, in: IEEE INFOCOM, 2005.
[36] H. Choi, P. McDaniel, T. La Porta, Privacy preserving communication in MANETs, in: IEEE SECON, 2007.
[37] A. Kate, G. Zaverucha, U. Hengartner, Anonymity and security in delay tolerant networks, in: SecureComm, 2007.
[38] R. Sakai, K. Ohgishi, M. Kasahara, Cryptosystems based on pairing, in: Symposium on Cryptography and Information Security, 2000.
[39] D. Boneh, M. Franklin, Identity based encryption from the weil pairing, in: SIAM Journal of Computing, 2003.
[40] R. Jansen, R. Beverly, Toward anonymity in delay tolerant networks: threshold pivot scheme, in: MILCOM, 2010.
[41] A. Shamir, How to share a secret, Commun. ACM, 1979.
[42] J.R. Douceur, J.S. Donath, The sybil attack, in: IPTPS, 2002.

[43] X. Lu, P. Hui, D. Towsley, J. Pu, Z. Xiong, Anti-localization anonymous routing for delay tolerant network, Elsevier COMNETS, 2010.
[44] R. Friedman, A.-M. Kermarrec, H. Miranda6, L. Rodrigues, Gossip-based dissemination, in: MiNEMA, 2009.
[45] A. Lindgren, A. Doria, O. Schelen, Probabilistic routing in intermittently connected networks, Lecture Notes in Computer Science, 2004.
[46] X. Wang, S. Chen, S. Jajodia, Network flow watermarking attack on low-latency anonymous communication systems, in: IEEE Symposium on Security and Privacy, 2007.
[47] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: IEEE Symposium on Security and Privacy, 2007.
[48] M. Pirretti, P. Traynor, P. McDaniel, B. Waters, Secure attribute-based systems, J. Comput. Secur. (JCS), 2010.
[49] V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in: ACM CCS, 2006.
[50] A. Seth, U. Hengartner, S. Keshav, Practical security for disconnected nodes, in: Workshop on Secure Network Protocols (NPSec), 2005.
[51] E.P.C. Jones, L. Li, P.A.S. Ward, Practical routing in delay-tolerant networks, in: ACM SIGCOMM Workshop on Delay-Tolerant Networking, 2005.
[52] A. Keränen, J. Ott, T. Kärkkäinen, The ONE simulator for DTN protocol evaluation, in: SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques, ICST, New York, NY, USA, 2009.

Cong Shi is a Ph.D student at College of Computing, Georgia Institute of Technology. He received his B.S. and M.S. degrees (with honors) in computer science from Shanghai Jiaotong University (China) in 2005 and 2008, respectively. His research interests cover networked systems and network security and privacy.



Xiapu Luo is a research fellow at the Hong Kong Polytechnic University. He received his Ph.D. from the same university in 2007. He was a postdoctoral research fellow at the Georgia Institute of Technology. His research focuses on network security and privacy and network measurement.

Patrick Traynor is an Assistant Professor in the School of Computer Science at the Georgia Institute of Technology, and is also a member of the Georgia Tech Information Security Center (GTISC). He received his Ph.D. from The Pennsylvania State University in 2008. In addition to serving on a number of program committees, he is also a member of the Editorial Board for the Encyclopedia of Cryptography and Security. His research is focused in areas including telephony security and provenance, security for mobile phones and the systems issues associated with applied cryptography.

Mostafa H. Ammar is a Regents' Professor with the College of Computing at Georgia Tech. He has been with Georgia Tech since 1985. He received the S.B. and S.M. degrees from the Massachusetts Institute of Technology in 1978 and 1980, respectively and the Ph.D. in Electrical Engineering from the University of Waterloo, Ontario, Canada in 1985. His research interests are in network architectures, protocols and services. He has contributions in the areas of multicast communication and services, multimedia streaming, content distribution networks, network simulation and most recently in disruption tolerant networks. He served as the Editor-in-Chief of the IEEE/ACM Transactions on Networking from 1999 to 2003. He is a Fellow of the IEEE and the ACM.

Ellen W. Zegura received the B.S. degree in Computer Science (1987), the B.S. degree in Electrical Engineering (1987), the M.S. degree in Computer Science (1990) and the D.Sc. in Computer Science (1993) all from Washington University, St. Louis, Missouri. Since 1993, she has been on the faculty in the College of Computing at Georgia Tech. She received tenure and promotion to Associate Professor in 1999. She was promoted to Full Professor in 2004. She is currently Associate Dean and Chair of the Computing Science and Systems Division. Her research area is computer networking, with interests in content distribution, wide-area (Internet) topology and routing, and mobile wireless networking. She served as Editor-in-Chief of IEEE/ACM Transactions on Networking from 2003 to 2005, and she has served on the program committees of numerous networking conferences.