Discount Anonymous On Demand Routing for Mobile Ad hoc Networks

Liu Yang
Software Engineering College
Sichuan University
Chengdu, 610065, China
Email: yangliutww@gmail.com

Markus Jakobsson
School of Informatics
Indiana University Bloomington
Bloomington IN 47408, USA
Email: markus@indiana.edu

Susanne Wetzel
Department of Computer Science
Stevens Institute of Technology
Hoboken NJ 07030, USA
Email: swetzel@cs.stevens.edu

***Abstract*— Recent years have seen a large number of proposals for anonymity mechanisms operating on the application layer. Given that anonymity is no stronger than its weakest link, such proposals are only meaningful if one can offer anonymity guarantees on the communication layer as well. ANODR – or ANonymous On Demand Routing – is one of the leading proposals to deal with this issue. In this paper, we propose a novel technique to address the same problem, but at a lower cost. Our proposal, which we dub Discount-ANODR, is built around the same set of techniques as ANODR is. Our proposal has the benefit of achieving substantially lower computation and communication complexities at the cost of a slight reduction of privacy guarantees. In particular, Discount-ANODR achieves source anonymity and routing privacy. A route is "blindly generated" by the intermediaries on the path between an anonymous source and an identified destination. Route requests in Discount-ANODR bear strong similarities to route requests in existing source routing protocols, with the limitation that intermediaries only know the destination of the request and the identity of the previous intermediary – but not whether the latter was the originator of the request. The response to a route request protects the compiled route by means of iterated symmetric encryption, drawing on how messages are prepared before being submitted to a typical synchronous mix network (or onion router). The communication of data subsequently uses such "route onions" to channel the packet to the intended destination. We do not use any key exchange, nor do we utilize public key operations at any time; consequently, we do not need to rely on any PKI, CRL or related constructions.**

I. Introduction

Mobile Ad hoc NETworks (MANETs) are being used in a large array of settings in which there is no "hardwired" network infrastructure. Commonly cited uses include military applications, emergency rescue and disaster relief, however MANETs are also believed to have future uses within vehicular networking, manufacturing, and various types of surveillance and monitoring applications. While the last few years have witnessed substantial efforts to provide efficient [17], [10] and secure [9], [11], [8], [12] communication in ad hoc networks, much less emphasis has been placed on privacy issues.

Privacy (and the lack thereof) is one of society's most notable vulnerabilities. The attacks arising from insufficient privacy – whether of actions, identities or locations – can be of economic nature (whether for identity theft or targeted advertising); may relate to national security (by ways of attacks on the infrastructure and key individuals); and can affect personal security and mobility. Most recently, we have seen a remarkable upswing of privacy intrusions driven by attempts to perform identity theft. It is evident that location information may be used to better target victims of such attacks, as well as attacks in the entire spectrum mentioned above. To limit the success of such attacks – without having to re-engineer our entire communication infrastructure – it is important to develop techniques that implement sufficient levels of privacy, without demanding substantial changes of the network or the computational requirements associated with performing routing.

The motivation for this paper is to design a lightweight privacy-preserving on demand routing protocol to achieve *source anonymity* and *routing privacy*. We define source anonymity as a property guaranteeing that an adversary can not find evidence that a node is the originator of an observed message or route request. Our definition of routing privacy corresponds to a property of hiding the identities of nodes on a path, from an adversary who may control one or more of these intermediaries.

Our approach is based on reactive source routing, where a route is obtained only when there is a demand to send a message. Reactive routing is believed to have less overhead than proactive routing, where all routes are maintained periodically. Our solution is very efficient in that the most computationally intensive operation in use is symmetric encryption, no key exchange is used, and no public key operations are needed. For an ordinary application with eight hops, our solution increases the overall time by less than 0.2 ms(compared to DSR [13]), which approximately corresponds to a 3.53% increase. When using our approach to send messages, on a path of eight hops, the ratio of control bytes to packet size is approximately 7.3%, slightly higher than the value of 3.91% by DSR. Our solution is also easy to integrate into an existing system with little modification and low overhead.

A. Our contribution

Our main contribution is to design a lightweight, privacy-preserving routing protocol using only symmetric key operations. As such, this protocol is particularly well-suited for ad hoc networks with battery-constrained nodes.

1-4244-0423-1/06/$20.00 ©2006 IEEE

In our protocol, when a node wants to learn a path to a destination, it locally broadcasts a route request, which will be propagated to the destination node over one or more hops. Once the destination node receives the request, it responds with a route reply to the node it received the request from. An intermediary receiving a reply first appends its identity. Then it chooses a symmetric secret key to re-encrypt the message. The result is sent to the next intermediary on the way to the initiator of the request. Finally, the initiator obtains a route reply encrypted layer-by-layer, like an *onion*, by all the intermediaries on the route. The discovered route is stored in the *route cache*. When a source intends to send a message to a destination, it selects the corresponding onion from the route cache, appends it to the payload, and forwards the assembled packet to the next hop according to the information in the route cache. An intermediary learns the next hop by peeling off its layer of the onion using its secret key. It is important to note that our protocol does not require any key exchange. In fact, a node can choose its secret key arbitrarily.

B. Outline

We begin by describing our work in general and covering some related work in Section II. We then detail our communication model and adversary model in Section III. In Section IV, we describe the protocols for route discovery, message sending, and route maintenance. Privacy properties and the cost of the protocol are analyzed in Section V and VI. Finally, we conclude in Section VII.

II. Related Work

Privacy-related communication has applications on the Internet [6], [7], [19], [18], [21], ad hoc networks [15], hybrid networks [3], and sensor networks [14]. Some solutions focus on sender anonymity [4], [5], [6], [7], [19], [20], [21], other schemes provide recipient anonymity [22] still others provide both [1], [15], [3].

Chaum proposed a so-called *mix* to achieve source anonymity [4]. A mix accepts a number of messages from different sources, performs cryptographic transformations on the messages, and forwards the messages to the next destination in a random order. Mixes make tracking a particular message either in bit-pattern, size, or ordering with other messages difficult. Onion routing is an early and independent implementation of the notion of a mix network on the Internet by Goldschlag, Reed, and Syverson [6], [7], [19]. In onion routing the initiator of a connection creates an onion, defining the path of the connection through the network. Each onion router along the path learns its successor and some other information by peeling off one layer of the onion with its private key, and the data arrives at the destination in plaintext. Mix and onion routing are based on public cryptography. In order to build onions defining meaningful routes the onion proxy is required to know the network topology and public certificates of routing nodes. Crowds [20] is a web-oriented anonymizing approach for synchronous communications. The main difference between Crowds and mix-based solutions is that the routing path and length in Crowds is dynamic, not like the static or preset paths in mix-based solutions.

Kong and Hong proposed an solution, named ANODR, for anonymous on demand routing in mobile ad hoc networks [15]. Our approach bears some similarity to the onion construction of ANODR, but is different in the following ways. First, ANODR uses public cryptography to exchange the pseudonym for each hop en route, but our approach only employs symmetric building block. Second, each route discovery in ANODR causes a global onion-construction flooding (all directions) in the networks, and each node receiving the route request tries to open a trap-door function by performing a symmetric decryption and a comparison operation. If a node is not the destination, it will add another layer to the received onion by performing a symmetric encryption, and then broadcast the updated onion. In our approach, onions are constructed in only one direction – on the return route. Third, the trap-door design in ANODR requires a key distribution process, and each sender must know the trap-door key of the recipient in order to start a route request. Our approach does not require any key distribution. Finally, a potential drawback of ANODR is *lack of terminating condition*, i.e., large amount of requests will be propagated in the network for a long time. Traditional routing protocols solve this problem by setting a maximum propagating limit to each request [13]. However, a hop counter tells the adversary how far he is away from the originator. Hence it does not work in a protocol aiming at privacy preserving. We address the problem of surviving onions in a simple and practical way in this paper.

While ANODR achieves good privacy by providing *sender anonymity*, *recipient anonymity*, and *unlinkability*, it has a much higher overhead in computation, communication, and storage than our approach due to the above operations. In particular, for a route of eight hops, ANODR adds around 1.6 seconds over a conventional routing protocol for each route discovery, while our approach only increases 0.17 millisecond to perform such a route discovery. It is not clear whether ANODR is applicable in cases of frequent communications (among users) and high mobility due to it takes a long time to find route. ANODR can be used in cases requiring high level privacy, e.g., military and intelligence. Our approach is more efficient and provides acceptable privacy.

Čapkun, Hubaux, and Jakobsson design an approach by changing node pseudonyms and cryptographic keys to achieve privacy in hybrid ad hoc networks [3]. They make each node be able to sign a verifiable message, but no one can tell who signed it except the central authority. The approach in [3] can not be applied to ad hoc network where there is no central authority. Like other public-key oriented protocols, their approach achieves privacy by adding a huge computation burden and complexity to the network.

Blaze, Ioannidis, and Keromytis et al, proposed WAR (Wireless Anonymous Routing) in [2]. WAR works in cases where all nodes are in direct communication range of each other. It relies on public cryptography and has an expensive key distribution mechanism. Although periodically sending

symmetric keys is not time consuming, the receivers need to decrypt these keys with their private keys. Another problem of WAR is there is no route discovery. A source selects intermediaries to a destination at random, and encapsulates the payload by an iterated encryption approach. It is hard to imagine that a randomly selected route will be a valid path to a destination, given that the source has no knowledge of the topology of the network. Thus, a source channels message to a destination in a *blindly-flooding* way. Unlike WAR, our approach works in multi-hop situations and has a clear way to perform route discovery.

Waters, Felten, and Sahai introduce the "incomparable" public keys to achieve receiver anonymity [22]. In their work, a receiver is able to create many public keys corresponding to one secret key. Since a receiver has many public keys, he is able to generate many identities. An adversary, given two public keys, is not able to tell whether they are equivalent, i.e., corresponding to the same secret key. The incomparable public key approach can be used as a building block to achieve receiver anonymity in wired or wireless routing. Unlike [22], our approach does not aim for receiver anonymity.

Our approach distinguishes itself from the other approaches by two features: it is based on symmetric cryptography; and we do not use any key exchange, nor do we utilize public key operations at any time.

III. Problem Statement

An important feature of multi-hop ad hoc networks is that nodes collaborate to realize communication. In particular, when a node wants to communicate with a destination not in its direct communication range, it requires the assistance of one or more intermediaries to forward its message; or in other words, it needs to learn a path to the destination. DSR [13] and AODV [16] are two well-known reactive routing protocols. Unfortunately, they do not provide privacy protection for either route discovery or communication. In both DSR and AODV, a route request identifies the source and destination and the hop-count the request has traveled so far. A route request of DSR even records the sequence of nodes as it is propagated to the destination. A route response of DSR includes all intermediaries between the source and destination in plaintext. In AODV, each entry in the route table contains the number of hops to different destinations.

There are a number of ways to compromise the privacy of DSR and AODV. For example, an adversary is able to learn the identities of all intermediaries by just eavesdropping in DSR. Long time eavesdropping allows him to learn the topology of the network. Eavesdropping also allows an adversary to learn the topology of the network in AODV.

A. Communication Model

We assume the network to be a mobile ad hoc network. Nodes in the network can either be devices with regular computation ability, like laptops, or lesser computation ability, like pocket PCs (PDAs), or devices with low computation and storage ability, like sensors. Nodes are distributed uniformly in the network. All nodes are willing to cooperate in the network operation. Nodes do not exchange local topology information with their neighbors who are within direct communication range. A message is propagated from the source to the destination over one or more hops. Each node in the network has a unique identity, and it chooses an arbitrary secret key which is not shared with any other node. The communication between two nodes is bidirectional, i.e., if node A can reach node B, then B can reach A.

B. Adversary Model

1) Assumption: We let $\mathcal{A}$ be an adversary controlling one or more nodes in the network. We assume:

- The nodes controlled by an adversary are not able to monitor all nodes within the direct communication range of the source.
- $\mathcal{A}$ can be *passive* or *active*. A passive adversary obtains information only by eavesdropping, while an active one may post route requests, inject messages, tamper with, or even drop received messages to gain information. We assume an adversary has bounded eavesdropping ability. In particular, $\mathcal{A}$ is able to eavesdrop on no more than a fraction ϵ of nodes en route.
- We also distinguish compromised nodes by *en route*, *close* to a route, or *far away* from a route. Close refers to a compromised node is within one hop from a given route, and far away means that a compromised node is more than one hop from a given route. A node en route or close to a given route is able to hear both the route request and the corresponding response to a certain route discovery. A node far away from a route can not hear the corresponding route response.

2) Goals of the adversary: There are two goals of an adversary $\mathcal{A}$. The first goal is to learn the identity of a message originator. The second goal is to infer identities of all nodes along a route. If an adversary achieves both the first and the second goals, he is naturally able to associate two nodes as communication partners.

We say a protocol achieves *source anonymity* if $\mathcal{A}$ is not able to find evidence that a node was the originator of a specified message. Moreover, we say a protocol provides *route secrecy* if $\mathcal{A}$ can not attain the second goal with a probability non-negligibly higher than a random guess from a group of candidates. Furthermore, we say a protocol provides *pair-anonymity* if $\mathcal{A}$ can not link two nodes as communication partners from routing evidence. This feature follows automatically from the first two properties.

3) Possible attacks: An adversary may use several methods – either passive or active – to achieve one or both of the goals described above.

- *Tracing*: an adversary tries to trace the route by controlling one or more compromised nodes close or en route.
- *Timing attacks*: an adversary tries to guess its distance to the source or destination by observing timings associated with the repeated use of same onions.

- *Onion recording*: an adversary deduces the source or route length by recording a lot of onions.
- *Tampering*: a node manipulates messages to gain information.
- *Packet dropping or injecting*: a node drops packets or injects packets to the network.

IV. Solution

To simplify the description of our approach, we introduce some denotation:

- rid: route request id;
- S, D: source and destination;
- $E_{K_i}()$, $D_{K_i}()$: symmetric encryption and decryption with secret key K_i;
- REQ: route request in form of (rid, S, D);
- REP: route reply in form of (e_i, n_i, rid), where $e_i = E_{K_i}(e_{i-1}, n_{i-1})$, and n_{i-1} is the next hop to D.

A. Intuitive approach

We use an example to explain our main idea. Figure 1 and 2 demonstrate the procedure of route discovery by a node S. First, S initiates a route discovery by assembling a route request $REQ = (rid, S, D)$ and broadcasts it locally. The REQ is heard by n_1, a neighbor of S. n_1 then helps to propagate the request by flipping a biased-coin. If the result of flipping is true, it replaces the S in REQ by its own identity n_1 and broadcasts the request locally; otherwise it drops the request. Other nodes deal with the received request in the same way. The request is transmitted several hops until it is heard by the destination D. D then assembles a route response $REP = (e_D, D, rid)$ (shown in Figure 2) and transmits the REP to n_5, the sending neighbor of the corresponding route request. n_5 encrypts the received (e_D, D) by its secret key K_5 and sends (e_5, n_5, rid) to n_4, from which n_5 received the route request with rid. The route response is encrypted layer by layer by intermediaries n_4, n_3, n_2 and n_1 when traveling from D to S. Each layer of the encryption contains an identity of the next intermediary to D. Finally, S receives a path to D in form of (e_1, n_1, rid). It then adds an entry (D, e_1, n_1) to its route cache. Figure 1 shows the big picture of our approach, and Figure 2 shows the details of route discovery. For simplicity, we omit two things in all further figures: the coin-flipping operation, and a flag used to indicate whether a message is a route request or response.

B. Route discovery

The process of route discovery includes the following three stages:

1) Request origination: A source S initiates a route request to a destination D by selecting a rid and locally broadcasting a route request $REQ = (rid, S, D)$.

2) Request resolution: The process of request resolution is shown in Figure 3.

Each node maintains a $PROCESSED_{REQ}$ to record the rids already processed in order to avoid repeated transmission of the same requests. A node n_i also maintains a list L_i

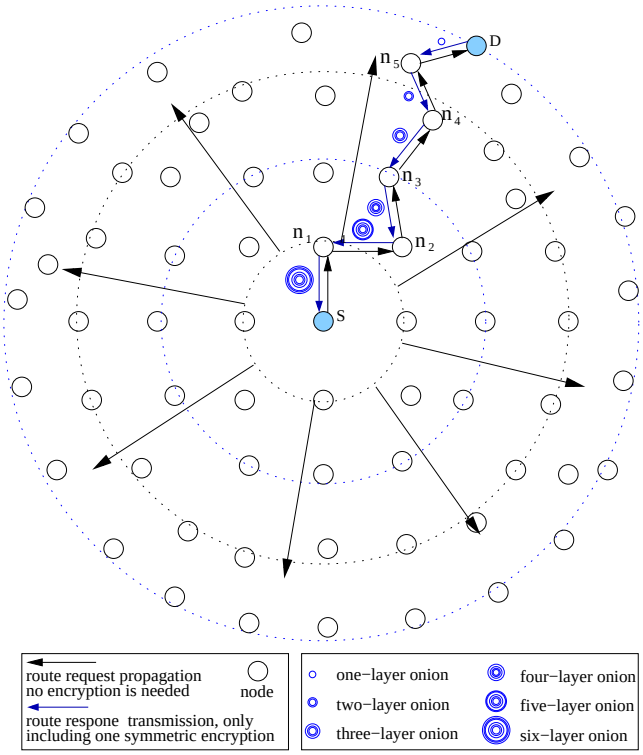


Fig. 1. An example of route discovery by our approach

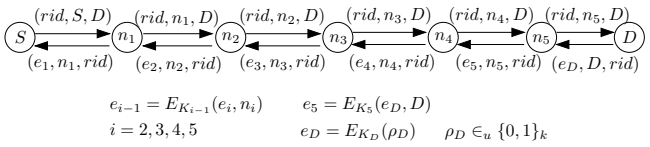


Fig. 2. The process of forming a privacy route from S to D by symmetric encryption.

containing entries of (S_i, rid), where S_i is a neighbor from which n_i received a request with rid. L_i will be used later for route reply propagation. A destination constructs a route reply by encrypting a binary string ρ_D with random length between 0 to k bits, where k can be adjusted according to the network situation. The random length here makes the size of the reply somehow unpredictable. Each node maintains a route cache containing entries in the form of $(destination, path, nexthop)$. A node calls $Retr(D, cache)$ to retrieve an entry from its route cache according to destination parameter D. The result of the "call" can either be $null$ or in the form of $(path, nexthop)$, where $path$ is an encrypted route. We note that the propagation of route requests will continue for a long time if not controlled. DSR uses a hop counter (recording how many hops a request has traveled) to solve this problem. A route request will be dropped if its hop counter is greater than some upper limit. Such an approach can not be used here because a hop counter tells an observer how far he is from the request originator. To overcome this problem, we let each node flip a biased-coin (denoted by $Flip(bcoin)$) to determine whether to retransmit a request. For example, if

A node n_i receives a control message (rid, S_i, D) from S_i.
If $rid \in PROCESSED_{REQ}$, then
    % ignore repeats
    halt;
else put (S_i, rid) in list L_i;
If $D = n_i$, then
    % recipient is destination
    respond $REP = (e_i, n_i, rid)$ to S_i, halt;
    where $e_i = E_{K_i}(\rho_D)$, and K_i is the secret chosen by n_i.
else if $Retr(D, cache) \neq null$, then
    %recipient knows the path
    respond $REP = (e_i, n_i, rid)$ to S_i, halt;
    where $e_i = E_{K_i}(path(D), nexthop)$,
    and $(path(D), nexthop) = Retr(D, cache)$
else if $Flip(bcoin) = true$
  % retransmit by flipping biased-coin
    locally broadcast (rid, n_i, D), halt;
else
    halt.

Fig. 3. Steps of route request resolution

the winning rate of the biased-coin is $\frac{9}{10}$, then a request is expected to be retransmitted 5 hops on average before being dropped. The winning rate of the flipping should be adjusted according to the network situation such that route requests will visit every node with reasonable probability. This depends on several factors, like the number of nodes, average hops between communication pairs, etc. Too high of a winning rate may cause a traffic jam in the network, while too low of a value will cause frequent failure of route discovery.

We note that whether a node is performing "request origination" or "request resolution" is indistinguishable to an observer (whether a neighbor or an adversary) as long as the observer is not able to monitor all neighbors of this node. The reason for this is that a request only contains the identity of the sending neighbor, which can either be an originator or be an intermediary.

3) Response transmission: The steps of routing response transmission are shown in Figure 4.

A node n_j receives a $REP = (e_i, n_i, rid)$.
if $rid \in PROCESSED_{REP}$, then
    % ignore repeats
    halt
else if $rid \in MYREQ$, then
    % n_j is the initiator
    % add an entry to route cache
    add (D, e_i, n_i) to route cache, halt;
else if $Retr(rid, L_j) \neq null$, then
    % n_j is an intermediary
    assemble $REP = (e_j, n_j, rid)$ and send it to S_j,
    where $S_j = Retr(rid, L_j)$, and $e_j = E_j(e_i, n_i)$
otherwise
    % invalid REP
      discard REP.

Fig. 4. Steps of response transmission

Each node maintains a $MYREQ$ to record the rids of route requests it has issued. If it receives a REP with an rid belonging to $MYREQ$, it knows this reply corresponds to a request it has issued and adds an entry (D, e_i, n_i) to its route cache. If a node n_j receiving a reply is not the corresponding originator, it checks whether it is an intermediary by calling a function $Retr(rid, L_j)$. Here L_j is a list containing entries of (S_i, rid), S_i is a neighbor from which n_j received a request with rid. This approach allows the reply to reach to the request originator. If n_j finds an S_j in L_j corresponding to the rid in REP, then it is an intermediary. n_j then encrypts the (e_i, n_i) with its secret and sends the assembled REP to S_j.

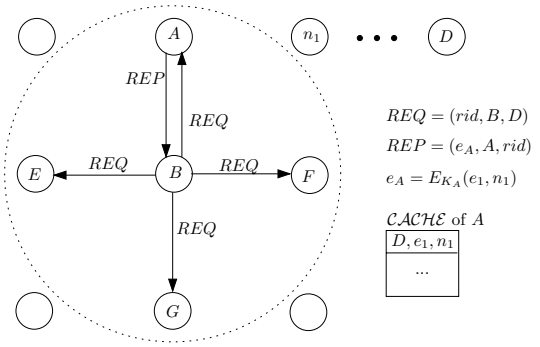


Fig. 5. An example of using cache to speed up the route discovery

Figure 5 is an example of using route cache to speed up the route discovery process. Node B issues a route request REQ to D by doing a local broadcast. The request REQ is heard by its neighbors A, E, F, and G. Node A retrieves its cache by D and gets an entry (D, e_1, n_1). It then assembles a route response $REP = (e_A, A, rid)$ and sends back REP to B, where $e_A = E_{K_A}(e_1, n_1)$. Hence B gets a route to D. We note E, F, and G will help to retransmit the REQ further because they are not the destination, and also have no entry to D in their cache. Later, another route response may be returned to B. This response can be used as a backup in case the first route is invalid.

C. Sending a message

Suppose S wants to send a message M to D. It first checks its route cache for an entry to D. If such a route is found, S can transmit M according to this route. Otherwise, S will start a route discovery to D and will get a route reply using the approach in Section IV-B. The process of sending a message can be described by the following protocol.

1) Message origination:

- S obtains an entry (D, e_i, n_i) from its cache or by means of route discovery.
- S assembles $P = (M, e_i, r_S)$ and transmits it to n_i, the next hop on the route to D. Here $r_S = E_{K_S}(\rho_S)$ and $\rho_S \in_u \{0,1\}^k$.

Here ρ_S is a 0-1 string chosen by S with a random length between 0-k bits, and r_S is the most inner layer of the route from the D to S. We note e_i can be only used to route messages from S to D due to its encryption order. Therefore,

another route needs to be built in case D wants to send a reply message to S. Such a route can be generated in the same way as e_i except in an inverse encryption order, which starts from S and ends up at D.

2) Message transmission: The steps of message transmission are shown in Figure 6.

A node n_j receives a packet $P = (M, e_j, r_{j-1})$ from n_{j-1}.
n_j computes $(e_{j+1}, n_{j+1}) = D_{K_j}(e_j)$
if $D_{K_j}(e_j) \in G_j(\rho)$
    % n_j is the destination
    accept M, halt;
else if n_{j+1} is a neighbor
    transmit (M, e_{j+1}, r_j) to n_{j+1}, where $r_j = E_{K_j}(r_{j-1}, n_{j-1})$.
otherwise
    % invalid packet
    discard P.

Fig. 6. Steps of message transmission

Node n_j maintains a list $G_j(\rho)$ to keep track of the route replies it resolved before. If the decrypted part of a route e_j can be found in $G_j(\rho)$, then n_j is the destination of message M; otherwise it just helps re-transmit the message to the next hop indicated by the decrypted result.

Figure 7 is an example of sending a message, where the message M, originated from S, travels through node n_1, n_2, n_3, n_4, and n_5 to arrive at the destination D. We see that a return route r_5 has been established by the intermediaries when M arrives at D. This returning route is used to send back an acknowledgement or a message to the originator when necessary.

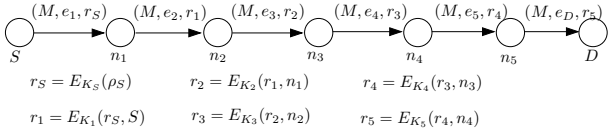


Fig. 7. S sends a message to D via a privacy route

The construction of a return path r_i occurs during the first message from S to D after the route discovery. Once D learns the return path, it is unnecessary to perform such a construction during the succeeding communication. This will reduce the overhead of control messages.

D. Route maintenance

We use an end-to-end scheme to maintain the route. After receiving a message from a source, D assembles an acknowledgement ack and appends the return route r_i, then it sends back the acknowledgement to the hop from which it received the original message. The ack will be propagated back to the source, where each intermediary removes a layer of the return route by decrypting the route with its secret key. Figure 8 shows an example, where D responds with an ack to S via route r_5 after receiving a message. If the source does not receive an ack in a certain time interval, it assumes a route error occurred. It may choose another route or start a new route discovery to the destination.

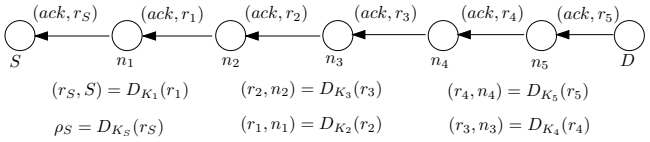


Fig. 8. D sends back an acknowledgement to S via a privacy route after receiving a message

E. Responding to a message

If the destination needs to send back a message to the source, it uses the same approach as that in sending back an acknowledgement (Section IV-D). We note that when an acknowledgement or a responding message is being sent back to the source S, the destination D and each intermediary are unaware of the identities of the source and nodes other than its predecessor and successor along the route.

V. Privacy Properties

Our protocol has the benefit of achieving substantially lower computation and communication complexities at the cost of a slight reduction in privacy guarantees. In particular, Discount-ANODR achieves source anonymity, routing privacy, and and pair-anonymity. We show that route privacy is well preserved if less than half of nodes en route or close to a route are compromised. Intermediaries only know the destination of the request and the identity of the previous intermediary, but not if the latter was the originator of the request. To some extent, our protocol also protects the locations of nodes in the network when protecting their identities. In the following statements, we analyze our protocol by considering possible attacks and comparing its privacy features and overhead to other protocols.

A. Preventing Attacks

1) Tracing: To our knowledge, message *tracing* is the most powerful of all attacks. It is possible for an adversary to trace a route by controlling several compromised nodes en route or close to a route. If two nodes are observed transmitting the same message, they are two hops along a certain route. The adversary is able to learn all intermediaries if he can control enough number of nodes en route or close to a route. We note as long as a compromised node is close to a certain route, it does not matter whether it is an intermediary because it is able to eavesdrop on passing messages.

If all nodes along a route are disclosed by the adversary, we say the route is fully traced; otherwise the route is said to be partially traced, or un-traced (no node en route is traced). Here we define a metric – *trace ratio* – as the fraction of hops being traced en route.

$$R = \frac{\sum_{i=1}^{l} c_i}{L} \tag{1}$$

where l denotes the number of compromised segments of a certain route, c_i is the length (number of hops) of the ith compromised segment, and L is the length of a route.

Fig. 9. A route with eight hops

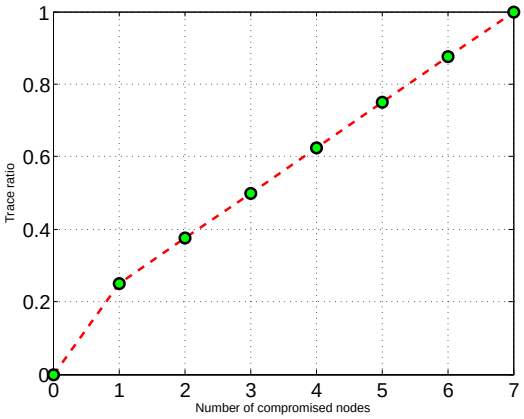
We demonstrate the trace ratio of our protocol by investigating a route with 8 hops (Figure 9) and consider three cases.

1) **Consecutive colluding:** nodes compromised by $\mathcal{A}$ are consecutive en route or close to a route. For example, if node n_2 is compromised, then $\mathcal{A}$ learns two hops (n_1 to n_2, and n_2 to n_3), the trace ratio is computed as $\frac{2}{8} = \frac{1}{4}$; similarly, if n_2 and n_3 are compromised, the trace ratio is $\frac{3}{8}$. Figure 10(a) shows the trace ratio corresponding to different numbers of consecutive compromised nodes en route or close to a route. In this case, $\mathcal{A}$ needs to compromise at least 7 consecutive nodes to trace the whole route. However, even with that information, $\mathcal{A}$ can not identify the source with certainty.
2) **Two hops distance:** the distance between nodes compromised by $\mathcal{A}$ is two hops. For example, if n_1 and n_3 are compromised, then $\mathcal{A}$ learns four hops en route, and the trace ratio is $\frac{4}{8} = \frac{1}{2}$. Figure 10(b) shows the trace ratio corresponding to different numbers of compromised nodes with two hops from one to another. To trace the whole route, $\mathcal{A}$ needs to compromise at least 4 nodes.
3) **Three hops distance:** the distance between compromised nodes is three hops. For example, if n_1 and n_4 are compromised, then $\mathcal{A}$ learns four hops en route, and the trace ratio is $\frac{4}{8} = \frac{1}{2}$. This is the same as case 2.

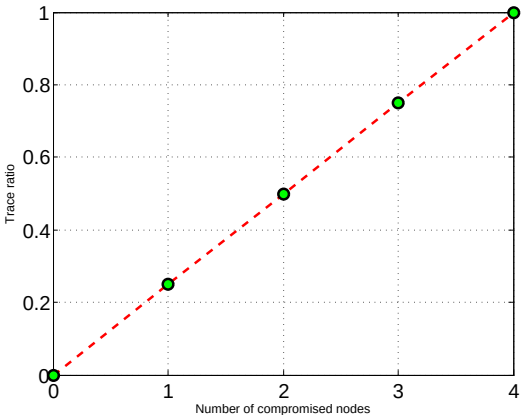
The above analysis and the results in Figure 10 show that the lower bound for $\mathcal{A}$ to fully trace a route requires $\mathcal{A}$ to compromise $\frac{1}{2}$ the number of nodes en route or close to a route. In other words, if less than $\epsilon = \frac{1}{2}$ of nodes en route or close to a route are compromised, the adversary is not able to fully trace the route.

2) Other attacks: Finally we discuss counter-measures to other attacks described in section III-B.

- *Timing attacks* can be thwarted by introducing a random delay before message re-transmission. Timing attacks do not help the adversary gain the identities of source or intermediaries.
- *Onion recording*: Repeated onions allow an adversary to learn that several messages might have been originated from a same source (the cached onion may be used by different nodes), but they do not identify the source. Such a threat can be reduced if we shorten the life-time of cached onions.
- *Tampering*: An adversary may tamper with a route request, a route response, or an encrypted onion. Tampering with a route request may cause the request be dropped or be propagated to a wrong destination. Tampering a route response may cause an originator to not be able to receive the response, or to receive an invalid response. Tampering with an onion during message transmission will cause the next intermediary to not be able to decrypt the onion, and therefore drop the message. All these attacks do not help an adversary identify the source, or trace a route in use. The tampering can be prevented by enforcing authentication, which is an orthogonal topic to our paper.
- *Packet dropping or injecting*: When a message gets dropped, the source usually re-sends the message. If a node injects a route request to the network, it will receive a route response from a destination. If a node injects invalid onions to the network, the onions will be dropped by other nodes (unable to decrypt). Both packet dropping and injecting do not help an adversary to achieve his goals.



(a) Consecutive compromising



(b) Non-consecutive compromising

Fig. 10. Trace ratio corresponding to different types of colluding

B. Comparison of privacy features to other protocols

From analysis of Section III-B we see both DSR and AODV provide almost no privacy. For example, an eavesdropper en route or close to a route learns identities of all

nodes en route from a single intercepted message in DSR. AODV allows an active adversary to learn the topology of the network by repeatedly issuing route discovery. Our protocol achieves source anonymity and route secrecy. During the route discovery, an intermediary only knows the identity of the destination and the previous intermediary, but not whether the latter was the request originator. The route constructed by iterated-encryption protects the identities of intermediaries. To fully trace a route, an adversary has to compromise at least half the number of nodes en route or close to the route. We note ANODR [15] provides better privacy than our protocol, but with a much higher overhead in computation, storage, and communication.

VI. Overhead

A. Overhead in route discovery

We base our estimate of the overhead of DSR, our protocol and ANODR on the following facts and assumptions:

- Nodes are uniformly distributed in the network, and the average travel distance of a route request is 10 hops.
- For ANODR, the field length of source, destination, and route pseudonym is 128 bits. The length of the other nonce is 40 bits. Let the average size of onions be 600 bits. For our protocol, the length of the identifier of a node is 40 bits, which is the same as the nonce in ANODR. The size of an acknowledgment is 40 bits.
- When public-key encryption is used, we assume the use of ECAES (160-bit key), with an encryption time of 160ms, and decryption time of 42ms. When symmetric encryption is used, we assume the use of AES (128-bit key & block). The encryption/decryption speed of AES is 29.1/29.2Mbps. The above computation time is based on iPAQ3670 pocket PC with Intel StrongARM 206MHz CPU.
- The maximum packet size for the three protocols is 1500 bytes. The channel capacity is 2 Mbits/second. (Windows systems are limited to a maximum payload size of 1380 bytes for TCP packets.)

1) Route discovery time by DSR: The route request by DSR has the format $REQ = (rid, S, rrcd, D)$, where $rrcd$ is the route record used to keep the sequence of hops traveled as the request is propagated through the network. The route response has the format $RES = (rid, rrcd)$. According to the assumptions above, for a path with 8 hops, the route discovery time can be estimated to be 4.82 milliseconds.

2) Overhead of ANODR and our approach: We compare the overhead of computation, storage, and communication between our approach and ANODR for a route discovery with a length of 8 hops. The computation overhead refers to how much time is spent on public-key and symmetric-key encryptions and decryptions; storage overhead refers to how much space is required to keep secrets, public/private keys, lists, and pseudonyms, etc; communication overhead refers to the amount of data being sent through the network. Using the above assumptions, the computation time of ANODR is approximately 1.6 seconds, while in our approach it is only 0.17 milliseconds. Our approach increases the route discovery time with 3.54% compared to DSR, while ANODR takes significantly longer time (more than 330 times that of DSR) to perform a route discovery. The storage costs for ANODR and our approach are estimated to be approximately 12.7Mb and 62kb respectively. The communication costs of ANODR and our approach are 417kb resp. 81 kb.

By analysis, we identify two reasons responsible for the big difference in costs between our approach and ANODR. One is that ANODR adopts public cryptography, but we do not. Another reason is related to the use of onion encryption. Onions are encrypted on the way out in ANODR [15], i.e., from the request initiator to every direction in the network, while the onion encryption occurs only on the returning way in our approach.

To understand the big difference between ANODR and our protocol, it is worth to take a look at the two approaches in more detail. Figure 1 and 11 show the processes of route discovery from S to D in the two protocols. In Figure 1, a route is constructed on the return path by D and intermediaries n_5, n_4, n_3, n_2 and n_1. Each intermediary adds one layer to the received onion. Onions only occur on the return path. However, in Figure 11, the onion is a part of a route request. As the route requests travel away from S, onions get bigger and bigger (more layers). They are propagated in all directions from S without any explicit terminating condition. On the return path, each hop "costs" two public-key operations, two symmetric-key operations (one is used to peel off one layer of an onion), and one comparison. These operations add a heavy computational burden to nodes on the route.

We will now take a look at the costs of the WAR [2] protocol in a network with the same topology as in Figure 1 and 11. In WAR, S selects multiple intermediaries at random. Such randomness is just like no route discovery or "blind" flooding. Given that S has no knowledge of the network topology in this multi-hop environment, S can not find a route to D.

B. Control bytes in packet transmission

We use the same assumption as in Section VI-A to estimate the length of control bytes of the protocols when sending a message, i.e., 8 hops route, 600 bits onion on average, 128 bits route pseudonym. The comparison of control ratios with DSR, ANODR, and our approach is summarized in Table I.

TABLE I

Comparison of length of control message via packet size between different protocols when sending messages

Protocol	Control message (byte)	R_c=Control msg/Packet size		
		512 packet	1024 packet	1500 packet
DSR	40	7.81%	3.91%	2.67%
Discount ANODR	150 (first packet)	29.30%	14.65%	10.00%
	75 (succeeding packet)	14.65%	7.3%	5.00%
ANODR	16	3.13%	1.56%	1.07%

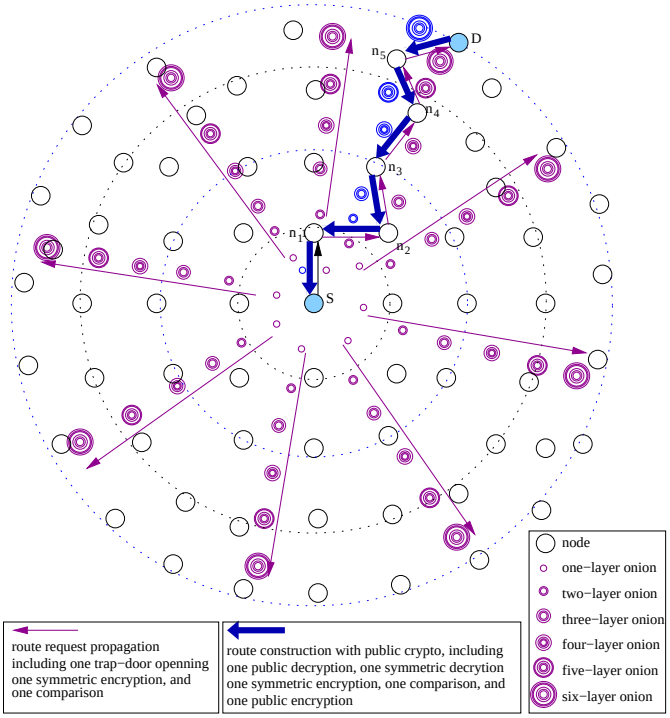


Fig. 11. An example of route discovery by ANODR.

The control-to-message ratio of ANODR is smaller than that of both DSR and our protocol in the packet transmission phase. However, it is easy to see that the overhead is shifted to the route discovery in ANODR. In an 8-hop route discovery, the communication overhead is approximately 52125 bytes, while it is only 10125 bytes in our protocol.

Table I shows how the control-to-message ratios decrease for an increasing packet size. In our approach, the ratio for the first packet is high due to the construction of the return path. It gets lower in the succeeding communication. We recommend to use 1024 bytes as the packet size in our protocol.

VII. Conclusion

We propose an approach – *Discount ANODR* – for anonymous on demand routing in mobile ad hoc networks. We provide peer-to-peer privacy of both payload and control messages using a cryptographically lightweight protocol relying solely on symmetric cryptography for its operation. As a result, we achieve substantially lower computation and communication complexity in comparison with functionally related proposals, at the cost of only a minor reduction of privacy guarantees. Effectively, however, the achieved reduction of the burden borne by the user devices is believed to enable the actual deployment of a privacy preserving technique of this type; thus, we argue that we actually *enhance* privacy guarantees (in comparison to the status quo) as opposed to degrading them. Our proposal achieves source anonymity and routing privacy. As long as less than half of the nodes close to or on a given route are compromised, an adversary will be unable to trace a route. The overhead analysis indicates our approach increases the route discovery time over DSR for a typical application by less than an estimated four percent, making our protocol particularly suitable to use in ad hoc networks with high mobility. In terms of future work, we plan to quantify the relationship between the biased-coin and the traffic reduction of the request flooding versus the probability of route discovery. We also plan to extend our work to BGP on the Internet.

Acknowledgments

The authors thank Dr. Xiaofeng Wang for his helpful discussion on the related work, and for Jared Cordasco for helpful feedback on the format of the article. We also appreciate the reviewers' helpful comments and suggestions.

References

[1] L. Ahn, A. Bortz, and N. J. Hopper. k-anonymous message transmission. In *Proceedings of the 10th ACM conference on Computer and Communications Security*, pages 22–130, Washington D.C., USA, 2003.

[2] M. Blaze, J. Ioannidis, A. D. Keromytis, T. Malkin, and A. Rubin. Protocols for anonymity in wireless networks. In *Proceedings of the 11th International Workshop on Security Protocols*, Cambridge, England, April 2003.

[3] S. Capkun, J. P. Hubaux, and M. Jakobsson. Secure and privacy-preserving communication in hybrid ad hoc networks. Technical report ic/2004/10, EPFL-DI-ICA, January 2004.

[4] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. In *Proceedings of Communications of the ACM, 24(2)*, pages 245–253, 1981.

[5] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[6] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[7] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private internet connections. *Communications of the ACM (USA)*, 42(2):39–41, 1999.

[8] Y. C. Hu, D. B. Johnson, and A. Perrig. Sead: Secure efficient distance vector routing for mobile wireless ad hoc networks. *Ad Hoc Networks Journal*, pages 175–192, 1 2003.

[9] Y. C. Hu, A. Perrig, and D. B. Johnson. Ariadne: A secure on demand routing protocol for ad hoc networks. In *MobiCom'02*, Atlanta, Georgia, USA, September 23-26 2002.

[10] Y. C. Hu, A. Perrig, and D. B. Johnson. Efficient security mechanisms for routing protocols. In *Proceedings of the Tenth Annual Network and Distributed System Security Symposium (NDSS)*, 2003.

[11] Y. C. Hu, A. Perrig, and D. B. Johnson. Packet leashes: A defense against wormhole attacks in wireless networks. In *IEEE Infocom*, 2003.

[12] Y. C. Hu, A. Perrig, and D. B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *ACM Workshop on Wireless Security (WiSe)*, 2003.

[13] D. B. Johnson and D. A. Maltz. *Mobile Computing*, volume 353, chapter Dynamic Source Routing in Ad Hoc Wireless Networks. Kluwer Academic Publishers, 1996.

[14] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk. Enhancing source-location privacy in sensor network routing. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, 2005.

[15] J. Kong and X. Hong. ANODR: ANonymous On Demand Routing with untraceable routes for mobile ad-hoc networks. In *ACM MOBIHOC'03*, 2003.

[16] C. Perkins. Ad-hoc on-demand distance vector routing. In *MILCOM '97 panel on Ad Hoc Networks*, Nov 1997.

[17] A. Perrigy, R. Canetti, D. Song, and J. D. Tygar. Efficient and secure source authentication for multicast. In *Proceedings of Network and Distributed System Security Symposium*, February 2001.

[18] M. Reed, P. Syverson, and D. Goldschlag. Proxies for anonymous routing. In *In 12th Annual Computer Security Applications Conference*, pages 95–104. IEEE, Dec 1995.

[19] M. Reed, P. Syverson, and D. Goldschlag. Anonymous connections and onion routing. *IEEE Journal of Selected Areas in Communications*, 16(4):482–494, May 1998.

[20] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1), June 1998.

[21] C. Shields and B. N. Levine. A protocol for anonymous communication over the internet. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*, Athens, Greece, 2000.

[22] B. R. Waters, E. W. Felten, and A. Sahai. Receiver anonymity via incomparable public keys. In *Proceedings of the 10th ACM conference on Computer and communication security*, 2003.