

# OLAR: On-demand Lightweight Anonymous Routing in MANETs

Yang Qin, Dijiang Huang, Vinayak Kandiah

Arizona State University, USA  
 {yang.qin.1, dijiang, vkandiah}@asu.edu

## ABSTRACT

To protect secure MANET communications, several anonymous routing schemes have been proposed to date. However, most previous approaches sacrifice networking performance due to heavy cryptographic operations. In this paper, we devise an On-demand Lightweight Anonymous Routing (OLAR) scheme, applying the secret sharing scheme based on the properties of polynomial interpolation. OLAR is an identity-free routing scheme, which provides source and destination anonymity, end-to-end communication relation anonymity, as well as route anonymity. In addition, the proposed anonymous routing scheme highly decreases the overhead of data transmission, while making packets more untraceable compared to the previous solutions. The performance of OLAR is demonstrated by experiments and comparison.

**Keywords:** Anonymous Routing, Lightweight, Secret Sharing, MANET

## 1 Introduction

Wireless stations in a MANET communicate with each other by broadcasting signals to whoever listens to the wireless channel. It is easy for attackers to eavesdrop on the messages. Moreover, a MANET is a non-infrastructure network. Every node within the network can be both an end-user and a router and it may move frequently. If the attackers could insert or compromise a node in the network, they obtain the same rights as all the other legitimate nodes. In this case, security requirements are obviously tougher than usual.

In hostile environments, communication anonymity has to be preserved to ensure security. Basically, secret messages must not be seen by unauthorized parties. Network members should not reveal their real identities to others but use pseudonyms (*identity-free*). Their locations also need to be concealed to protect themselves (*location anonymity*). The source and destination of a flow should be indistinguishable among all the nodes (*source/destination anonymity*) and their end-to-end relation should be hidden (*end-to-end communication relation anonymity*). A flow must be untraceable, and the route of a flow, i.e., all the forwarders in the path, should not be discovered by malicious adversaries (*route anonymity*). All these strict requirements become the heavy tasks of anonymous routing protocols. Fortunately, a variety of techniques have been proposed and utilized

as the countermeasures. Chaum designed a MIX network [3] for untraceable emails in 1981, which takes a notable role in recent developments of anonymous routing. In a MIX network, every message sender should pre-maintain the entire networking topology and randomly choose a path to transmit the message to the receiver. *Onion* structure [4][10] is another frequently used technique in MANET anonymous routing solutions. It provides anonymous connections that are strongly resistant to both eavesdropping and traffic analysis. Besides, tricks such as packet padding and shuffling, dummy traffic insertion are utilized for anonymous routing as well.

Previously proposed anonymous routing schemes like ARM[8], ASR[12], ANODR [5], MASK [11] and SDAR [2] have provided anonymity for MANET communications on certain levels, but they all suffer from different defects. The most common drawback is that they sacrifice the networking performance. Normally, a routing protocol has the route discovery phase and the message transfer phase. During the second phase, most of the prior schemes require the packets to be encrypted and decrypted at each node in the path. These cryptographic operations incur large cost. To cope with this problem, we propose a novel solution named On-demand Lightweight Anonymous Routing (OLAR) in this paper. OLAR takes the advantage of polynomial interpolation as Shamir's secret sharing scheme [9] does to assure the information privacy. During the route discovery phase, the source generates a polynomial for each path that is found by route requests flooding. Then the selected points on the curve which represents the polynomial are distributed to the forwarders on the path. With these points, the forwarders are able to forward messages to the destination without revealing the secret contained in it. The only operation they need to do is addition, instead of decryption and encryption. However, only the destination can get the original messages at the end. The main contribution of OLAR is two-fold:

- 1) OLAR improves the anonymity for MANET communications by making packets more untraceable, compared to the existing anonymous routing schemes;
- 2) OLAR avoids encrypting and decrypting packets at each hop during the message transfer phase, which notably decreases the overhead of data transmission i.e., OLAR is a "lightweight" routing scheme.

The rest of this paper is organized as follows: section 2 provides background information on existing MANET anonymous routing schemes. Section 3 introduces the

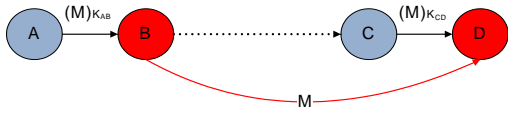


Figure 1: Defect of the Per-Hop-Encryption-Decryption Scheme. (Node B and D are compromised nodes. They can communicate with each other through a secret link. Comparing the original message (M) they see, they know they are on the same path. They are also able to infer that node A and C are on the same path as well.)

systems and models used in this paper. In section 4, we elaborate OLAR in detail and section 5 conducts the anonymity and performance analysis. We also demonstrate the performance of our solution through experiments in section 6. Finally, section 7 concludes our work.

## 2 Background

Previously proposed anonymous routing schemes like ANODR [5], MASK [11] and SDAR [2] have set up the basic framework of anonymous routing in MANETs, using different methodologies.

ANDOR is an on-demand routing protocol. The source who wants to initialize a conversation first broadcasts a route request (RREQ). The RREQ has an onion structure and a global trapdoor which can be opened by the destination only. Every node receiving the RREQ except the destination wraps the onion with its own secret and re-broadcasts it. When the destination receives the RREQ and opens the trapdoor, it generates and broadcasts a reply packet (RREP) with the onion structure from the RREQ embedded in the RREP. Through this process where the RREP is bounced back to the source, each intermediate node can share a secret with its successor and predecessor in the path. These shared secrets are then used as the keys to encrypt and decrypt the transmitted messages at each hop. These per-hop encryptions and decryptions change the message appearance at each hop. However, ANODR enables every node along the path to see the original messages. Thus, as shown in Fig. 1, if several compromised nodes on a same path collude with each other by comparing the original messages, they may not only find that they are on a same path but also figure out other nodes on this path.

Similar to ANODR, in the message transfer phase, MASK encrypts and decrypts packets at each hop based on the shared secret of each pair of adjacent nodes. However, these secrets are generated proactively during the anonymous neighborhood authentication. Before any routing operations take place on the routing layer, neighborhood authentication takes place at the MAC layer. Every pair of adjacent nodes use pairing-based cryptography [1] to share a master key, and then generate a group of shared secrets using the master key. The route discovery phase in MASK is only for constructing virtual-circuit-like paths for each end-to-end communi-

cation pair. To decrease the cost, a route request packet (ARREQ) in MASK has the destination’s real identity in plain text. Any legitimate node or attacker who receives the ARREQ will know who the destination is, i.e. MASK does not provide destination anonymity.

SDAR has a community key management system. Each central node proactively generates different kinds of keys for different nodes within its vicinity, based on their trust levels (High, Medium, and Low). To discover a path, the source sends out a request packet with a temporary public key (TPK) and its corresponding secret key (TSK). But the TSK is encrypted using the destination’s public key. Nodes which receive the request packet use TPK to encrypt their own information and append it to the packet. When the request packet arrives at the destination, the destination gets the TSK and in turn obtains the information of all nodes in the path. After that, the destination uses onion structure to make sure each pair of adjacent nodes along the path get a shared secret. These secrets are then used for normal message transmission as in ANODR and MASK. Nonetheless, a big problem is that the destination knows the identities of all the forwarders and their shared “secrets”. With this information, the destination can easily infer how far the source is, where it is and even who it is.

It is necessary to note that, all of the previous anonymous routing solutions perform per-hop encryptions and decryptions. These operations may make the packets more untraceable but significantly degrade the networking performance due to high computational overhead.

## 3 System and Models

In this section, we discuss the properties of polynomial interpolation which are used in Shamir’s secret sharing scheme [9]. Then we describe the adversaries’ abilities considered in this paper.

### 3.1 Polynomial Interpolation

We first present the basic idea of the polynomial interpolation based secret sharing scheme [9] and its properties which are utilized by our OLAR scheme. The operation of polynomial interpolation is over a finite field  $F$ . For the simplicity of presentation, in this paper, we assume that our description and presented examples use the field of characteristic zero.

The secret sharing mechanism conforms to a  $(k, n)$  threshold. The  $(k, n)$  threshold means that if the secret  $M$  is divided into  $n$  parts, then the knowledge of any  $k$  parts or more makes  $M$  easily computable but knowledge of any  $k - 1$  parts or fewer leaves  $M$  completely undetermined. The  $(k, n)$  threshold can be satisfied by Lagrangian polynomials. Given a set of  $k$  points  $(x_0, y_0), (x_1, y_1), \dots, (x_{k-1}, y_{k-1})$  where  $x_i \neq x_j$  ( $0 \leq i, j \leq k - 1$ ), a unique interpolation polynomial of degree  $k - 1$  can be

constructed as:

$$L(x) = \sum_{j=0}^{k-1} l_j(x) \cdot y_j, \quad (1)$$

where

$$l_j(x) = \prod_{i=0, i \neq j}^{k-1} \frac{x - x_i}{x_j - x_i}. \quad (2)$$

In, (1) and (2), the interpolation polynomial  $L(x)$  is in Lagrangian form and  $l_j(x)$  is a Lagrangian basis polynomial. The polynomial cannot be constructed unless all the  $k$  points are acquired. For example, to share the secret  $M$  among  $k$  participants, we can select a polynomial  $L(x) = a_0 + a_1x + a_2x^2 + \dots + a_{k-1}x^{k-1}$  where  $a_0 = M$  and all the other coefficients are randomly selected values, i.e.,  $M = L(0)$ . Then we randomly select  $k$  points  $(x_0, y_0), (x_1, y_1), \dots, (x_{k-1}, y_{k-1})$  on the curve of  $L(x)$  and compute the corresponding Lagrangian coefficients  $l_j(0)$  ( $0 \leq j \leq k-1$ ). These values are distributed to the  $k$  participants, respectively, so that each of them gets a point and its Lagrangian coefficient. Only if all participants share their secrets (i.e., points), can they derive the secret  $M$  by using the following formula:

$$M = \sum_{j=0}^{k-1} l_j(0) \cdot y_j, \quad (3)$$

where  $l_j(0)$  can be obtained by substituting  $x = 0$  in (2).

## 3.2 Adversarial Abilities

In this paper, we assume the adversaries have the following basic abilities:

- 1) They can eavesdrop every message transmitted in the communication system.
- 2) They can collude with each other to exchange information and trace messages.
- 3) They can physically compromise a wireless node and acquire all the private information stored in this node. However, within a period of time, they can only compromise a limited number of nodes.

## 4 On-demand Lightweight Anonymous Routing

In this section, we describe the On-demand Lightweight Anonymous Routing in detail. The notations used in rest of this paper are given in table 1.

### 4.1 Basic OLAR Scheme (B-OLAR)

To elaborate the proposed solution clearly, we first describe a basic OLAR scheme, namely B-OLAR. B-OLAR is a simplified version of OLAR and has several defects, but it represents the basic ideas of OLAR. In Fig. 2, we presents a simple four nodes example to illustrate the B-OLAR protocol. Similar to other on-demand routing schemes, this scheme has two main phases: route discovery phase and message transfer phase.

$RPK_{src}$	one-time public key used by the source to uniquely identify an RREQ
$RSK_{src}$	the corresponding private key of $RPK_{src}$
$SN$	a random number generated by the source
$GPK_{dest}$	the global public key of the destination
$GSK_{dest}$	the corresponding private key of $GPK_{dest}$
$TPK_A$	temporary public key of node $A$
$TSK_A$	the corresponding private key of $TPK_A$
$K_{SA}^i$	a shared key with source $S$ in node $A$ 's key pool
$N_A^i$	a nonce randomly generated by node $A$
$b_i$	Lagrangian coefficient of point $(x_i, y_i)$
$(M)_k$	a message $M$ encrypted using the key $k$

Table 1: Notations

#### 4.1.1 Anonymous Route Discovery

In order to initialize a conversation with the destination  $D$ , the source  $S$  broadcasts a route request (RREQ). The initial RREQ has the following format:

$$\langle RREQ, RPK_{src}, (RPK_{src}, SN)_{GPK_{dest}}, TPK_S \rangle .$$

*RREQ* is the message type (route request).  $RPK_{src}$  is a one-time public key of the source node, which is changed in every RREQ sent by the source. Thus, each RREQ has a unique  $RPK_{src}$ , which can be used as its identifier. When broadcasting an RREQ, the source stores the  $RPK_{src}$  and its corresponding private key  $RSK_{src}$  in its *route request table*.  $SN$  is a randomly generated number. It is wrapped together with  $RPK_{src}$  by the destination's global public key as a global trapdoor [6]. Only the destination can open the trapdoor and acquire  $SN$ , which is necessary for the destination to get authenticated by the source. Finally,  $TPK_S$  is a temporary public key of the node forwarding the RREQ. Initially, it is a temporary public key selected by  $S$ .

After node  $n1$  receives the RREQ, it checks whether it is a duplicated packet by looking up the  $RPK_{src}$  in its key mapping table. If the RREQ has been received previously, the message is simply discarded. Otherwise,  $n1$  tries to open the trapdoor using its own private key. If it fails,  $n1$  replaces  $TPK_S$  with a temporary public key  $TPK_{n1}$  and broadcasts the following message:

$$\langle RREQ, RPK_{src}, (RPK_{src}, SN)_{GPK_{dest}}, TPK_{n1} \rangle .$$

In addition, node  $n1$  also fills in its *key-mapping table* with  $RPK_{src}$ ,  $TPK_S$ , and  $TSK_{n1}$ . These values will be used to set up a virtual circuit when  $n1$  receives a reply message from the destination.

Eventually, when  $D$  receives the RREQ and opens the trapdoor using its private key  $GSK_{dest}$ , it has to compare the  $RPK_{src}$  in  $((RPK_{src}, SN)_{GPK_{dest}})$  with that in plain text to verify the integrity of the RREQ. If these two keys are different,  $D$  ignores this RREQ; otherwise,

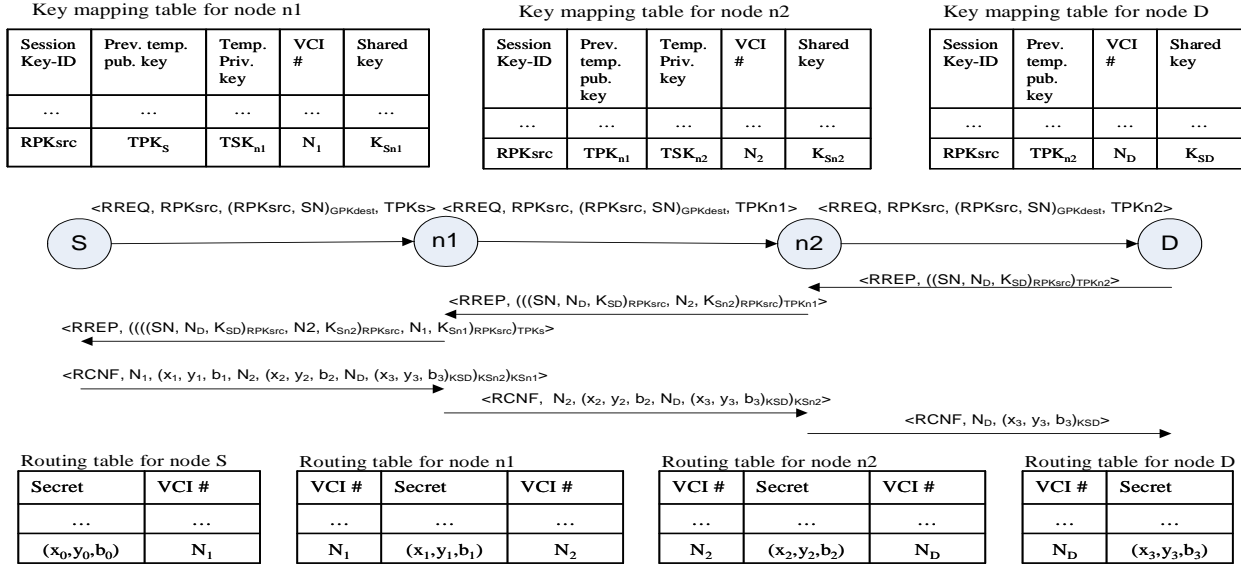


Figure 2: B-OLAR protocol.

$D$  replies a route reply message (RREP) with the following format:

$$\langle RREP, (SN, N_D, K_{SD})_{RPK_{src}} \rangle_{TPK_{n2}} \cdot \quad (4)$$

In (4), RREP indicates the message type (a route reply).  $N_D$  is a randomly generated nonce which is later used as a virtual circuit identifier (VCI). A symmetric key  $K_{SD}$  is also randomly selected to share with the source. The message is encrypted by  $RPK_{src}$ , so that only the source can decrypt it, and then it is encrypted by node  $n2$ 's public key  $TPK_{n2}$ . To keep track of the generated nonce and corresponding keys,  $D$  stores  $N_D$  and  $K_{SD}$  in its key mapping table. In addition,  $D$  also creates an entry in its routing table indexed by the VCI number  $N_D$  (shown in Fig. 2). Once the RREP is received by  $n2$ , it decrypts the message using its temporary private key  $TSK_{n2}$ . Then,  $n2$  generates a random nonce  $N_{n2}$ , selects a shared key  $K_{Sn2}$ , appends them to the onion, and encrypts the onion with  $RPK_{src}$ . Finally,  $n2$  looks up the temporary public key of its predecessor ( $TPK_{n1}$ ) in the *key-mapping table* and encrypts the onion using  $TPK_{n1}$ . The RREP sent by  $n2$  has the following format:

$$\langle RREP, ((SN, N_D, K_{SD})_{RPK_{src}}, N_{n2}, K_{Sn2})_{RPK_{src}} \rangle_{TPK_{n1}} \cdot$$

The above mentioned procedure will continue until the reply message reaches the source (see the packet information on each hop in Fig. 2). When the source receives the RREP, it first decrypts the message using  $TSK_S$  and then uses  $RSK_{src}$  to strip the onion layer-by-layer. Eventually, the source will retrieve all the shared keys and VCI numbers as well as the number  $SN$ . If the destination gets authenticated successfully by providing the

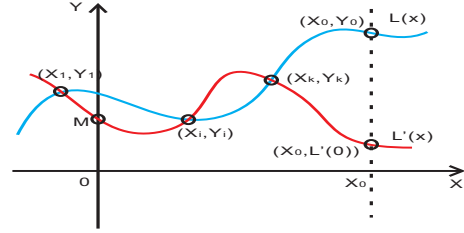


Figure 3: Curve Construction

correct  $SN$ , the source will then create a route confirmation message (RCNF).

Before sending an RCNF message, the source  $S$  generates  $k + 1$  points on a polynomial of degree  $k$ , say  $L(x)$ , as shown in Fig. 3, where  $k + 1$  equals the total number of nodes on the path from  $S$  to  $D$ . Then the source calculates the Lagrangian coefficients  $b_0, b_1, \dots, b_k$  for these points using:

$$b_j = \prod_{i=0, i \neq j}^k \frac{x_i}{x_i - x_j}. \quad (5)$$

Note that  $b_j = l_j(0)$  in (2). The source  $S$  creates and sends a RCNF packet with the following format:

$$\langle RCNF, N_1, (x_1, y_1, b_1, N_2, (x_2, y_2, b_2, N_D, (x_3, y_3, b_3)_{K_{SD}})_{K_{Sn2}})_{K_{Sn1}} \rangle \cdot \quad (6)$$

RCNF is the message type (route confirmation) and the rest of (6) is an onion structure. In this onion, each nonce  $N_i$  is followed by the information encrypted using a pairwise shared key  $K_{S^*}$ , where  $*$  represents  $n1, n2$ , or  $D$ . The pair  $(N_i, K_{S^*})$  is generated by the  $i_{th}$  successor of the source along the route to the destination and the last pair  $(N_3, K_{SD})$  belongs to  $D$ . Therefore, each node

receiving the RCNF can determine if the packet is for itself by checking its key mapping table. Based on the decrypted message, each node can fill in its routing table with incoming VCI number and outgoing VCI number. For example,  $n1$  can decrypt the message and build an entry in its routing table as shown in Fig. 2, where  $N_1$  is the incoming VCI number,  $N_2$  is the outgoing VCI number, and the tuple  $(x_1, y_1, b_1)$  is the secret used in data transfer. In this way, each node can build up its own routing table.

#### 4.1.2 Anonymous Message Transfer

Based on the path set up in the route discovery phase, messages can be transferred from the source to the destination anonymously. Suppose the source  $S$  has a message  $M$  destined to node  $D$ . For example,  $M$  can be an integer, which can be mapped to a sequence of characters based on publicly known transformation settings. During the route discovery,  $k + 1$  points  $(x_i, y_i)$  (where  $0 \leq i \leq k$ ) on the curve formed by a randomly constructed polynomial  $L(x)$  have been selected by  $S$ . Each node on the path is distributed one of them and its corresponding coefficient  $b_i$ . The point  $(x_0, y_0)$  and its coefficient  $b_0$  are retained by  $S$  itself. However, as shown in Fig. 3, to send the message  $M$  to the destination through this path, the source  $S$  needs to construct another curve of degree  $k + 1$ , which is determined by the original  $k$  points  $(x_i, y_i)$  ( $1 \leq i \leq k$ ) and a new point  $(0, M)$ , i.e., the polynomial of the new curve, say  $L'(x)$ , must satisfy  $L'(0) = M$  and  $M$  is the secret. In addition, the source  $S$  updates its own point by making  $y'_0 = L'(x_0)$ , which means the new point  $(x_0, y'_0)$  is also on the new curve. According to (3), we can derive:

$$M = y'_0 b_0 + \sum_{i=1}^k b_i \cdot y_i, \quad (7)$$

where  $b_0$ ,  $b_i$  and  $y_i$ ,  $1 \leq i \leq k$ , are unchanged and retained by each node along the path. Therefore, to transfer the message  $M$  to the destination anonymously, the source creates the following data packet:

$$\langle DATA, N_1, y'_0 b_0 \rangle .$$

DATA denotes the message type (data),  $N_1$  is the VCI number to identify next hop (note that only  $n1$  knows  $N_1$ ), and  $y'_0 b_0$  is the initial message created by the source. When  $n1$  receives the message, based on its routing table, it will send the following message to  $n2$ :

$$\langle DATA, N_2, y'_0 b_0 + y_1 b_1 \rangle .$$

Then  $n2$  will send the following message to  $D$ :

$$\langle DATA, N_D, y'_0 b_0 + y_1 b_1 + y_2 b_2 \rangle .$$

Eventually,  $D$  will recover the message  $M = y'_0 b_0 + y_1 b_1 + y_2 b_2 + y_3 b_3$ .

#### 4.1.3 Defects in B-OLAR Scheme

B-OLAR is just a simplified version of OLAR. The defects of B-OLAR are presented as follows:

1. In a RREP packet, each node on the path between the source and destination adds exactly one layer to the onion structure, so the source has the ability to determine the length of the path. Consequently, by requesting to set up a route, the source can discover how many hops there are from the source to the destination. That means, a malicious inside node can easily estimate the distance from every other node to itself by sending RREQ to each node, and in turn acquire the topology information of the network.

2. In B-OLAR, every node will add constant value to the payload. Each node along the path retains a point  $(x_i, y_i)$  on the curve and its corresponding Lagrangian coefficient  $b_i$ . When a node in the path receives a data packet, it adds the value of  $b_i \times y_i$  to previously received payload  $M'$ . Since the values of both  $b_j$  and  $y_j$  are constant for each intermediate node (and the destination), the difference between incoming and outgoing messages transferred along the same route will be constant at each hop for all data transmissions. This is a good indication to correlate incoming messages and outgoing messages to/from a node. This defect of B-OLAR leaves a chance to the attackers to identify a communication path.

## 4.2 OLAR

The OLAR protocol solves the problems of the basic scheme described above by applying the following mechanisms.

#### 4.2.1 Multi-points For Each Node

In the B-OLAR scheme, each node en route only gets one point on the curve. OLAR allows each node to have multiple points. The number of points retained by a node is decided by itself without being released to the source. To obtain  $k$  points from the source  $S$ , the destination  $D$  should reply the RREQ using the an RREP as follows:

$$\langle RREP, ((((((SN, N_D^1, K_{SD}^1)_{RPK_{src}},$$

$$N_D^2, K_{SD}^2)_{RPK_{src}}, \dots, N_D^k, K_{SD}^k)_{RPK_{src}})_{TPK_A} \rangle .$$

In this RREP packet,  $D$  randomly generates  $k$  pairs of nonce and shared keys and adds  $k$  layers to the onion structure. Each layer is encrypted with  $RPK_{src}$  so that when the RREP is received by  $S$ ,  $S$  cannot determine if these  $k$  pairs of nonce and shared keys belong to one node or multiple nodes. As shown in B-OLAR,  $D$  needs to store these nonce and shared keys in its key mapping and routing tables for later use. Similarly, each intermediate node also adds several layers to the onion structure.

Upon receiving the RREP,  $S$  performs exactly the same operations as in B-OLAR. However, in this case, the degree of the generated polynomial (the curve)  $L'(x)$

is no longer determined by the number of nodes en route but by the total number of nonce (or shared keys). Each nonce acquires one point from the source. When an RCNF created by the source arrives at an intermediate node on the path, the packet may appear like the following:

$$\begin{aligned} < RCNF, N_A^1, ((x_i, y_i, b_i), \\ & N_A^2, ((x_{i+1}, y_{i+1}, b_{i+1}), \\ & \vdots, \\ & N_A^n, ((x_{i+n-1}, y_{i+n-1}, b_{i+n-1}), \\ & N_B^1, (\dots)_{K_{SB}^1} K_{SA}^n \dots)_{K_{SA}^2} K_{SA}^1 > \end{aligned}$$

Node  $A$  searches its key mapping table and finds the item  $N_A^1$  as well as its corresponding shared key  $K_{SA}^1$ . The shared key is then used to decrypt the outer layer of the onion and get the information about the first point. This procedure keeps going until the nonce on top of the onion is not generated by  $A$ . That is, when  $A$  realizes  $N_B^1$  is not generated by itself,  $A$  sends out the packet and keeps all points it deserves. In addition,  $A$  updates its routing table as shown in table 2:

...	...	...
$N_A^1$	$(x_i, y_i, b_i)$	$N_B^1$
$N_A^1$	$(x_{i+1}, y_{i+1}, b_{i+1})$	$N_B^1$
...	...	...
$N_A^1$	$(x_{i+n-1}, y_{i+n-1}, b_{i+n-1})$	$N_B^1$
...	...	...

Table 2: Routing table of node  $A$  in OLAR

During the message transfer phase, if  $A$  receives a data packet as:

$$\langle DATA, N_A^1, M' \rangle,$$

it will update the value of  $M'$  using:

$$M' = M' + \sum_{j=i}^{i+n-1} b_j \cdot y_j, \quad (8)$$

and replace  $N_A^1$  with  $N_B^1$ , then send the updated packet.

#### 4.2.2 Different Increments For Different Messages

In the B-OLAR scheme, messages traveling through a node for the same communication session will be increased by the same value. To avoid this defect, OLAR changes the Lagrangian coefficient values  $b_j$  for every message. According to equation (5),  $b_j$  changes only if the  $x$ -coordinates change. Thus, unlike what is shown in Fig. 3 ( $x_0$  is fixed), we have to change the  $x$ -coordinate of the point belonging to the source, i.e., each time the source sends out a new message, it must use a different point  $(x'_0, y'_0)$  by updating both  $x_0$  and  $y_0$ . Every intermediate node and the destination must be informed of

the new  $x$ -coordinate so that they can update their  $b_j$  values as:

$$b'_j = b_j \cdot \frac{x'_0}{x'_0 - x_j} \cdot \frac{x_0 - x_j}{x_0} \quad (9)$$

However, if the new  $x'_0$  is transmitted along the path without encryption, then even the eavesdroppers can find out the path by tracing this value, which undermines the route anonymity. Even if the value is encrypted, nodes on the path can collude to derive path information. To solve this problem, OLAR utilizes a global function  $f(x, C)$  for every node to get the new  $x$ -coordinate:

$$x'_0 = f(x_0, C), \quad C = 1, \dots, n. \quad (10)$$

$f(x, C)$  can be as simple as an incremental function like  $f(x, C) = x + C$  where  $C$  is a constant number, or a one way function. The function has the benefit that nodes do not need to remember previous computed value and  $C$  is a counter. In this way, nodes will keep on updating their  $b_j$  values using (10). It is necessary to distribute the original value of  $x_0$  to all the nodes along the path during the route discovery phase.

## 5 Anonymity and performance analysis

In this section, we analyze the OLAR protocol in terms of anonymity and performance issues.

1) OLAR ensures the privacy of messages transmitted in the network. Neither an intermediate node nor an eavesdropper can figure out the original messages. Only the source and destination can see them in plain text.

2) OLAR is an identity-free routing scheme. In both the route discovery and message transfer phases, real identities are not used. In contrast to MASK [11], to initialize a conversation, the source only has to know the global public key of the destination ( $GPK_{dest}$ ) but not its identity. Unlike normal distance vector routing protocols such as AODV [7], nodes do not reveal their real identities in a RREQ. One time public keys and shared keys are used instead. Without knowing the real identity of its successor along the reverse path, a node can still unicast a RREP, by encrypting the packet with the successor's temporary public key. Similarly, in a RCNF, the mark on top of each layer of the onion structure is not the real identity. Each node on the path is notified by the nonce generated by itself. The nonce are also used as the VCI's instead of real identities.

3) The source/destination anonymity and end-to-end relation anonymity are protected by OLAR. Even the legitimate nodes along the path do not know who is the source or destination. In the route discovery phase, the source broadcasts a RREQ with its one time public key, but the direct neighbors have no way to know that the key belongs to the source. Similarly, when the destination broadcasts a RREP, the onion structure is encrypted layer by layer using the route public key ( $RPK_{src}$ ). Thus, the neighbors of the destination cannot

find out that the packet comes from the destination directly. In the message transfer phase, nodes just forward the messages according to the VCIs. They cannot even guess how far the source or the destination is. Based on the source/destination anonymity, the end-to-end communication relations are also completely protected. Nobody can associate the source with the destination. Only the source knows the intended destination.

4) The route anonymity is preserved in OLAR. OLAR is actually a distance vector routing scheme, in which every node on the path does not have any information about the entire path. Furthermore, by applying a distributed polynomial interpolation mechanism, a message transferred along the path appears different at each hop. Attackers cannot figure out the path by tracing the messages. It is worth noting that, although most of the previous solutions (e.g., [5][11][2]) also make the appearance of a message changed from hop to hop, they achieve this using per-hop encryptions and decryptions. This kind of solution can only confuse the outside attackers but not the inside compromised nodes (see Fig. 1). However, OLAR does not allow any intermediate node to see the original messages, so that they are untraceable not only to the outside attackers but also to the intermediate nodes.

5) Compared to the previous solutions, the most important improvement of our design is that we significantly decrease the overhead. In the route discovery phase, an intermediate node only needs to perform constant times of encryptions and decryptions. The cost of the source is a little heavier since it has to construct curves, select points and calculate the Lagrangian coefficients in addition to the asymmetric encryptions and decryptions. However, in the message transfer phase, OLAR avoids using per-hop encryptions and decryptions. To forward the messages to their destinations, the forwarders only need to perform multiplications and additions, which incur much less overhead compared to traditional cryptographic operations.

## 6 Experiments

In this section, we estimate the performance of the proposed OLAR protocol with extensive experiments.

Firstly, we compare the computational overhead of AES encryption/decryption with that of numerical multiplication by benchmark testing. The hardware configuration of the machine used for the benchmark testing is: AMD 64 bit 2 Ghz processor with 960 MB RAM. The code for AES encryption/decryption using 256 bit keys in CBC and CFB modes is written in C++ and compiled in Microsoft Visual Studio 2005. The code to perform multiplications over a finite field is written in Maple 9. The time taken for performing a single AES decryption and a single multiplication over a field is calculated for various message sizes. These times in milliseconds are compared in table 3.

Msg size	AES 256 CBC/CFB modes (ms)	Multiplication (ms)
512	0.024	0.004
1024	0.047	0.006
2048	0.095	0.014
4096	0.181	0.040
8192	0.365	0.120
16384	0.720	0.383
32768	1.438	1.196

Table 3: Benchmark comparing AES and multiplication.

Secondly, we conduct a comparison study to demonstrate the lightweight performance of OLAR. In our simulation, there are 50 nodes randomly deployed in a  $1000 \times 1000m^2$  square area. The simulation time is 600s and Random WayPoint mobility model is used. We implement OLAR and ANODR[5] as the routing protocols. In OLAR, without losing generality, we assume that each node may obtain no more than 10 points for one end-to-end communication. For ANDOR, the improved version suggested in [5] is implemented, instead of its basic version which actually reveals the route pseudonyms to whoever listen to the channel. CBR traffic is generated between 10 end-to-end communication pairs. We simulate four scenarios with different mobility (in terms of the moving speed of nodes) and CBR traffic rates to perform comparative study, as shown in Fig. 4.

In Fig. 4, we present the results of average computational time per node in ANODR and OLAR for the four simulation scenarios. These figures are drawn based on the number of encryptions, decryptions, and multiplications, multiplied by the computational time provided in our benchmark as shown in Table 3.

The  $x$ -axis in the figure is increased by the message size. When the message size increases, based on the benchmark in Table 3, we can see that the time difference between multiplication and encryption/decryption operation first increases. However, when the packet size continues to increase, the difference decreases till a certain value. In Fig. 4(a) & (c), we also observe that this behavior is more prominent with low mobility. From the figures, when the mobility is high, the reconstructions of the path to the destination become frequent because the topology changes more frequently. Thus ANODR will incur more encryptions and decryptions. The figures show that ANODR consumes more encryption and decryption time when the moving speed is 10m/s, which confirms our analysis. We can see that OLAR exhibits better performance than ANODR when the message size is less than 8192 bits. When the traffic rate increases, obviously ANODR will perform more encryption/decryption operations. Although the number of multiplications in OLAR increases as well, we know from Table 3 that this time is less compared to encryption/decryption operations. In addition, the average computation time per

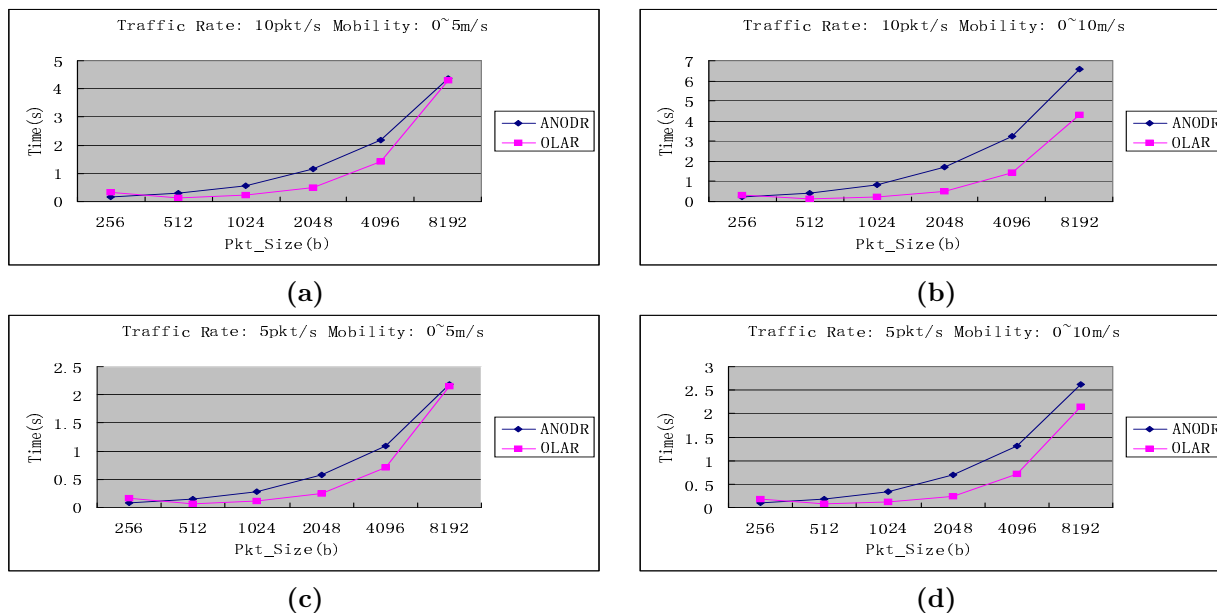


Figure 4: Time comparison of OLAR and ANODR.

node of ANODR increases faster than OLAR as the traffic rate increases. By comparing the figure 4 (a) with (c) and (b) with (d), we also observe this phenomenon.

## 7 Conclusion

Previously proposed anonymous routing protocols rely on per-hop encryptions and decryptions during message transferring to make the packets untraceable to outside attackers but they are not resistant to the insiders' attacks and also sacrifice the networking performance severely. To improve both the anonymity and the networking performance, we devise an On-demand Lightweight Anonymous Routing (OLAR) scheme. OLAR applies a distributed polynomial interpolation mechanism to achieve anonymous message transfer without per-hop encryptions and decryptions. The only task for a forwarder is to perform additions and multiplications, which cost much less than traditional cryptographic operations. Meanwhile, the messages traveling through a certain path have different appearances at different hops. Any forwarding node can not see the original messages as they are reconstructed only at the destination. This design makes the messages untraceable not only to outside attackers but also to malicious insiders.

## REFERENCES

- [1] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.
- [2] A. Boukerche, K. El-Khatib, L. Xu, and L. Korba. Sdar: a secure distributed anonymous routing protocol for wireless and mobile ad hoc networks. In

- Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, 2004.
- [3] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. In *Communications of the ACM*, 1981.
- [4] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private internet connections. In *Communications of the ACM*, 1999.
- [5] J. Kong and X. Hong. Anodr: Anonymous on demand routing with untraceable routes for mobile ad-hoc networks. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing*, 2003.
- [6] A. J. Menezes, P. V. Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
- [7] C. Perkins. Ad hoc on demand distance vector (aodv) routing, 1997.
- [8] S. Seys and B. Preneel. Arm: Anonymous routing protocol for mobile ad hoc networks. In *IEEE AINA*, 2006.
- [9] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.
- [10] P. Syverson, D. Goldschlag, and M. Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, 1997.
- [11] Y. Zhang, W. Liu, and W. Lou. Anonymous communications in mobile ad hoc networks. In *INFOCOM*, 2005.
- [12] B. Zhu, Z. Wan, M. Kankanhalli, F. Bao, and R. Deng. Anonymous secure routing in mobile ad-hoc network. In *29th Annual IEEE International Conference on Local Computer Networks*, 2004.