Certificate Free Anonymous Routing for Mobile Ad Hoc Networks

Amal Banerjee
Christine Julien
Vitaly Shmatikov

TR-UTEDGE-2005-005

Certificate Free Anonymous Routing for Mobile Ad Hoc Networks

Amal Banerjee[1], Christine Julien[1], Vitaly Shmatikov[2]
[1]Department of Electrical and Computer Engineering
[2]Department of Computer Science
The University of Texas at Austin
{abanerj@ece, c.julien@mail, shmat@cs}.utexas.edu

Abstract

This paper introduces a lightweight, anonymous, and secure routing protocol for mobile ad hoc networks. It leverages Identity Based Encryption (IBE) to provide a certificate-free key distribution mechanism. IBE is used only in the protocol's critical initial stage, to exchange cryptographic key material for subsequent operations. This minimizes computational cost and facilitates communication anonymity and data privacy. First and foremost, our protocol provides identity anonymity, which ensures that, although a node might determine that one of its neighbors is sending or receiving data, it is unable to pin-point the identity of that neighbor. Our protocol also provides route anonymity, which guarantees that a node forwarding a packet is unable to inspect the route, much less determine the original source or final destination. An added benefit of route anonymity is location anonymity—a node may suspect the presence of a node with a particular identity, but the node is unable to determine even the approximate physical location of the suspected node. The novelty of our approach is that our design goals are achieved using simple techniques, removing many of the compute intensive operations commonly associated with anonymous routing. In addition to anonymity guarantees, the protocol also ensures data privacy. This paper presents the details of our protocol, demonstrates its performance characteristics in comparison to alternative approaches, and analyzes its security properties.

1 Introduction

In mobile ad hoc networks, wirelessly enabled mobile nodes communicate opportunistically. Such networks are characterized by a complete lack of infrastructure and centralized administration. In these networks, each node must serve as a router for other nodes. These dynamic environments rely on node cooperation, and exchanging sensitive information requires an underlying routing algorithm to provide strong security guarantees. One desirable security guarantee is *anonymity*, which ensures that nodes participating in routing are not able to identify the communicating parties. For example, when a military commander uses a mobile ad hoc network to communicate with remote battlefield units, the exchanged information *and* the identities and locations of the communicating parties must be kept secret.

Anonymous routing requires establishing and maintaining routes under the condition that only the connected nodes can determine the other's identity (*identity anonymity*). In addition, a node's approximate location should not be revealed (unless explicitly desired by the node) (*location anonymity*). Finally, a successful protocol must guarantee that any forwarding node cannot determine the identity of any node on the route (including the sender and intended recipient) (*routing anonymity*). In addition, all data exchanged between communicating nodes must be completely concealed from routing nodes. These are demanding requirements in a mobile ad hoc network because of its inherent lack of infrastructure. Such requirements introduce limitations with respect to common approaches to routing where node identity information is integral to many protocols. Anonymity restrictions render many assumptions in mobile ad hoc routing invalid and necessitate novel approaches to providing anonymous communication.

Anonymous routing has been studied extensively in wired networks. These schemes have their roots in Mix-net [2], in which packets sent from a source to a destination must pass through a set of mixes. A mix re-orders and re-encrypts incoming data for forwarding, preventing correlation of incoming and outgoing flows. Mix-net was modified in Onion Routing [3], which allows routing information to be encoded in a set of encrypted layers, i.e., onions. Tor [4] (TCP based Onion Router) adds *forward secrecy* and incremental path-building. Recently, Mix-net has been extended to peer-to-peer networks [5]. In this adaptation, a source chooses a first mix from its set of nearest neighbors. In turn, the chosen mix selects another mix from its own set of nearest neighbors (excluding the source). Each time, the choice is based on an estimated *probability of forwarding*. Although these approaches provide a number of crucial anonymity guarantees, some drawbacks prevent their application to ad hoc networks.

In Mix-net, as the number of mixes increases, the number of re-orderings and re-encryptions also increases, making it increasingly difficult to correlate ingoing and outgoing messages. However, this also increases latency and decreases the delivery ratio. A mix node can also turn malicious and examine the contents of packets it re-encrypts. Onion Routing requires a central authority, making it impractical for mobile ad hoc networks. Tor requires a source to pre-negotiate a symmetric key for each hop on the route, which is virtually impossible in mobile networks due to node mobility. This is just a brief overview of anonymous communication; we compare our approach to a broad set of protocols in Section 6.

Our protocol targets anonymous routing directly to the needs of dynamic and unpredictable mobile ad hoc networks. Our approach leverages a combination of *node pseudonyms*, *Identity Based Encryption* (IBE), and simple symmetric key cryptography. Our use of IBE removes the burden of public key certificate management by using the identity of a node as its public key. The associated private key is known only by the node with the appropriate identity. This reduces computational cost because each route discovery does not have to include a public key certificate for verification by the intended destination. Our protocol also does not require persistent connectivity to a centralized authority, nor is there a need to establish pair-wise symmetric keys across an entire route. These features in combination make our algorithm more robust and lightweight in comparison to existing approaches. The unique contributions of our work are:

- Exploiting the power of an asymmetric key based cryptographic scheme while eliminating the use of public key certificates, and attendant certificate management infrastructure.
- Demonstration of the fact that seemingly compute expensive cryptographic operations can be used in a mobile ad hoc network, without actually incurring a high computational cost.
- Strong and lightweight anonymity guarantees (identity, routing and location) can be provided in a mobile ad hoc network by exploiting the fact that there are no links in a wireless network, which makes messages *untraceable*. A node receiving a packet is unable to determine who the sender is, unless the latter includes some identity information in a form that can be understood by the receiver, or the receiver examines the MAC layer information attached to the packet.

In the next section, we provide the assumptions and design goals underlying our protocol. Section 3 describes the protocol. We evaluate the protocol in terms of its performance (Section 4) and security implications (Section 5). Section 6 highlights related projects, and conclusions appear in Section 7.

2 Protocol Foundations and Design Goals

The next section details the low-level aspects of our protocol. First however, we carefully detail the basic assumption underlying our protocol and our specific design goals.

2.1 Basic Assumption

- A source knows only the identity (e.g., IP, URL, MAC address) of its intended destination, at the time it wants to establish the connection. The source does not know its neighbors' identities, and these neighbors do not know that the source is nearby, i.e., *no* node has a global view of the network.

With this assumption, the next section develops a protocol that provides anonymous routing and private data exchange. Specifically, our protocol satisfies several design goals that we outline next.

2.2 Design Goals
Our protocol provides three main anonymity guarantees and also ensures complete data secrecy between communicating pairs of nodes. The anonymity goals are:

- **Identity anonymity:** A node receiving or sending packets cannot be identified by its neighbors.
- **Route anonymity:** A node that forwards packets must be unable to determine the identities of other nodes that also participate in the routing protocol.
- **Location anonymity:** Even though a node might suspect that another node with a specified identity is present, it is not possible to determine even the node's approximate location.

Our target environment also necessitates that we ensure that a protocol is lightweight and robust. That is:

- Our protocol should limit the use of asymmetric key cryptography. This includes compute-intensive operations such as certificate verification and management.
- As route anonymity forbids node identities of routing nodes from appearing in clear text in any packet, route discovery and maintenance should use node pseudonyms to identify routes.
- The protocol should prevent flooding the network. This would adversely affect our protocol's performance *and* applications' performance since they rely on the network for data delivery.

Two common approaches to mobile ad hoc routing are table based [6, 7] and source routing protocols [8, 9]. We use source routing, primarily because table based schemes require nodes to constantly monitor information about their neighbors, and the computational burden incurred in managing this information while keeping the real identities of nodes secret is excessive. We present a source routing protocol that 1) prevents the need for persistent connectivity to a certificate authority and 2) takes advantage of both the power of asymmetric cryptography and the simplicity of symmetric key cryptography.

3 Certificate-free Anonymous Routing Protocol for Wireless Ad Hoc Networks (CARP)
In this section, we present our anonymous routing protocol for mobile ad hoc networks.

3.1 Protocol Overview
Our Certificate-free Anonymous Routing Protocol (CARP) achieves anonymous routing using a combination of light-weight techniques. A crucial problem is how cryptographic keys are created and distributed. Such concerns are important in mobile ad hoc networks, which lack fixed infrastructure. Many techniques use asymmetric cryptography to establish session keys but require expensive certificate management [18]. Other key establishment protocols use pair-wise symmetric key distribution schemes [10] but are highly probabilistic and unsuitable for practical ad hoc network applications. CARP circumvents these concerns using Identity Based Encryption (IBE) [1] during route discovery.

IBE is an asymmetric encryption algorithm that uses a node's identity as its public key. Due to the complexities associated with asymmetric-key cryptography, we limit IBE's use to route discovery. A route request encrypted with the destination's identity carries key material used to establish a *session key*, without requiring a source to present a public key certificate. The destination recovers the session key material using its private key. It sends a route reply containing additional material used to fully establish the session key. CARP uses source routing to incrementally build a route that includes the identities of the intermediate nodes. To provide route anonymity, CARP nodes substitute their actual identities with *pseudonyms*. Each packet carries a nonce that uniquely identifies it and prevents broadcast storms [12].

3.2 Node Data Structures
Before we describe our protocol, we first present the data structures that each node maintains.

3.2.1 Route Request Token Table
One of the basic data structures on each node is the route request token table. Each entry in this table contains a route request token, a two-byte nonce and a timestamp corresponding to the token's last

reference. Each token uniquely identifies a single route request packet, and this table is used to prevent excessive rebroadcast of a route request by a single node. As our protocol needs to be lightweight in order to conserve resources, the route request tokens that have not recently been used are removed.

3.2.2 Pseudonym Table

Another essential data structure is the list of pseudonyms a node uses to identify itself. Each pseudonym is unique to a route request that visits the node. Each entry in this table contains a pseudonym and a timestamp corresponding to when that pseudonym was last referenced. As in the route request token table, pseudonym values that have not recently been used in the protocol are removed from the table.

3.2.3 Routing data table

This table stores complete information for all active routes that exist between this node and all destination nodes. Table 1 lists the components of a single entry in this table.

destination id	the public identity of the destination node; this entry in the routing data table stores a route to this particular node
source node nonce	symmetric key material generated on this node and used as part of the session key material between this node and the destination referenced by the *destination id*
destination node nonce	symmetric key material generated by the destination and used as part of these session key material
initialization vector	used for encryption and decryption using the session key
encrypted initialization vector	the *initialization vector* encrypted with the session key; used as a trapdoor [13] to reduce decryption attempts (see Section 3.3)
pseudonym list	the list of node pseudonyms for this active route

Table 1: The components of an entry in the Routing Data Table

3.2.4 Route reply packet data table

This table stores information about recent route replies passing through this node. Maintaining this table ensures each packet reaches its intended destination by re-broadcasting it periodically until the node receives an acknowledgment. Until the entry is disabled, the route reply will be periodically rebroadcast using Truncated Binary Exponential Backoff [11]. Table 2 lists the components of an entry in this table.

route reply token	the token of the route reply packet this entry refers to; the entry persists only if it is not disabled
route reply packet uid	a set of bytes acting as a trapdoor; used to easily determine the intended destination
encrypted route reply data	data encrypted with the nonce sent by the source; contains a fresh nonce generated by the destination (for session key generation) and the source and destination identities (for helping the source map the reply to its initial request)
time_until_rebroadcast	time to hold this route reply before attempting rebroadcast
timestamp	time when this entry was inserted in the table.
rebroadcast_disabled	a true value indicates the route reply was re-broadcast by a neighbor
route pseudonym list	pseudonym list for this reply's route

Table 2: The components of an entry in the Route Reply Packet Data Table

3.2.5 Data packet table:

This table stores information about recent *data packets* that have passed through this node. The contents of the packets stored in this table are manipulated in a manner similar to that of the contents of the Route reply packet data table. Table 3 shows the components of an entry in the data packet table.

data token	unique token of the stored data packet.
data packet uid	the trapdoor for the data packet
encrypted data	application level data to be transmitted; encrypted with the session key generated cooperatively by the source and destination
timestamp	time when this entry was inserted
time_until_rebroadcast	time to hold this packet before attempting rebroadcast
rebroadcast_disabled	a true value indicates the packet was successfully forwarded
route pseudonym list	the pseudonyms that define the packet's intended route

Table 3: The components of an entry in the Data Packet Table

3.2.5 Route Request, Route Reply and Data packet structure

To ease the processing of control packets, CARP structures them in a common form:

[packet_type_, TTL, packet_token, other blocks of bytes that are specific to the packet],

where packet_type can be RREQ, RREP, or DATA for route request, route reply and data packets. The TTL (*time to live*) is decremented at each hop. The packet_token is a two-byte nonce that uniquely identifies each packet and is used to prevent broadcast storm [12].

3.3 Certificate-free Anonymous Routing Protocol (CARP)

Because Identity Based Encryption (IBE) is fundamental to CARP's functionality, we provide a brief overview. More details can be found in [1]. IBE is based on Elliptic Curve Cryptography [14] and the properties of pairing functions [15]. IBE uses a node's public identity (e.g., URL, IP address etc.,) as its public key, while the private key is generated by a key server using a master secret (known only to the key server) and public parameters of the elliptic curve. When a node is initialized (and only at that time), the key server provides it the parameters necessary to exchange encrypted material with other IBE nodes. This initialization can occur in many ways, for example by docking a device and interacting with a server online or by inserting a flash memory stick containing the information.

3.3.1 IBE Initialization

Before a node can participate in CARP, the key server must provide it the following IBE information:

- public elliptic curve parameters P and the product $s.P$. P is a publicly known point on the chosen elliptic curve, and s is a master secret that is known only to the key server.
- private key determined by the key server using the node's public identity and master secret s .

This information only has to be acquired a single time, and therefore CARP eliminates the need for persistent connectivity to a key management infrastructure.

3.3.2 Route Discovery

To initiate a route discovery, a source node hashes the public identity of the intended destination to a point on the elliptic curve and then generates:

- two large random numbers r and nonce_S
- a message $m_S = (ID_S, ID_D, \text{nonce}_S, \text{init\_vector})$ where ID_S , and ID_D are the identities of the source and destination. The init_vector is used by the subsequent symmetric key operations.
- the source generates the IBE encryption key K_{enc} using the destination's identity, the elliptic curve's parameters, and the *pairing function* [1, 14, 15] used by all nodes on the network.

The source encrypts m_S to get the cipher text $\{m_S\}K_{enc}$, using a symmetric encryption algorithm (e.g., Blowfish [16]) and calculates the product $r.P$. The source then creates a route request with the format:

$$[\text{RREQ, TTL, } r.P\text{ , route\_request\_token, } \{m_S\}K_{enc}\text{ , node\_pseudonym\_list}].$$

A node generates a unique route request token and a unique node pseudonym for each route request it initiates. It stores a copy of the former in its route request token table, and a copy of the latter in its pseudonym table. In addition, it stores the tuple

$$(ID_S, ID_D, nonce_S, \text{init\_vectorEnc})$$

in its routing data table, where init_vectorEnc is the init_vector encrypted with $nonce_S$. When a route is successfully constructed, the init_vectorEnc, (i.e., the uid) acts as a trapdoor [13] for subsequent packets. After the source broadcasts the packet, receiving nodes process and forward the request. Consider the small (static) network shown in Figure 1. We will use this simple network to demonstrate our protocol. Node A (e.g., the military commander from our scenario in Section 1) wishes to establish communication with node E (e.g., a battlefield unit). Node A must know node E's identity (e.g, IP address) in advance. Node A follows the process outlined above to create a route request packet, which, when broadcast, arrives at nodes B and C.

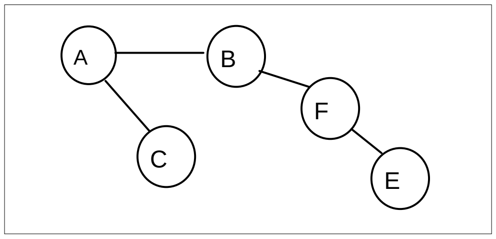


Figure – 1: A small ad hoc network.

3.3.3 Route Request Processing at an Intermediate Node

A node receiving a route request checks its list of pseudonyms for an element that matches any element in the packet's list. A match indicates that this node is already on the route and should not forward the packet again. If there is no match, the node compares the route_request_token against its list of request tokens. If there is a match, then the node has previously processed this request, and it is ignored. If not, the node attempts to recover the IBE encryption key K_{enc}, using its private key, $r.P$, and the elliptic curve parameters. *If the node successfully decrypts* $\{m_S\}K_{enc}$*, the node will extract its own identity from the clear text.* If the IBE decryption is unsuccessful, the node forwards a route request by:

- generating a unique pseudonym for this request, adding it both to its own table of node pseudonyms and the packet's node_pseudonym_list,
- copying the packet's route_request_token to its list of processed route request tokens, and
- decrementing the route request packet's TTL.

After visiting the other intermediate nodes e.g., node F in Figure 1, the route request packet reaches the intended destination.

3.3.4 Route Request Processing at the Destination.

The route request packet reaching the final destination has the contents:

$$[\text{RREQ, TTL, } r.P\text{ , route\_request\_token, } \{m_A\}K_{enc}, ran_S, ran_j, \ldots ran_i\ ],$$

where ran_i is the pseudonym node i added to the packet. At the destination, $\{m_S\}K_{enc}$ is successfully decrypted. The destination knows that the decryption is successful and that it is the intended destination because it finds ID_D in the clear text. In response, the destination generates and transmits a route reply.

3.3.5 Preparation for Route Reply

The destination generates a route reply $m_D = (ID_S, ID_D, nonce_D)$, where $nonce_D$ is generated by the destination. The node encrypts this packet with $nonce_S$ and init_vector to get $\{m_D\}nonce_S$.

3.3.6 Sending the Route Reply by destination

The destination creates a route reply packet with the following format:

$$[\text{RREP, TTL, route\_reply\_token, uid, } \{m_D\}nonce_S\text{, node\_pseudonym\_list}],$$

where node_pseudonym_list is copied from the request. The destination adds a unique pseudonym for itself to the list. The uid is init_vector encrypted with nonce_S and is a trapdoor that prevents a routing node wastefully attempting to decrypt the packet. The destination stores the tuple:

(ID_S , nonce_S , nonce_D , init_vector**,** node_pseudonym_list, uid)

in its routing data table. It then adds the tuple:

(route_reply_token, uid, $\{m_D\}\text{nonce}_S$, timestamp, time_until_rebroadcast, node_pseudonym_list)

to its route reply packet table. Until the destination receives a RREPACK (i.e., a route reply acknowledgement) from a neighbor, it will re-broadcast this packet at a period determined using Truncated Exponential Binary Backoff. When the destination receives a RREPACK, the packet is flushed from the table. En route to the source, the reply retraces the route established during discovery.

3.3.7 Route reply packet processing at any intermediate node

When a node receives a route reply, it checks its list of used pseudonyms for a match with any element in the packet's node_pseudonym_list. A match indicates that the node forwarded the corresponding route request and is part of the established route. In this case, the node processes the packet by first comparing the packet's token to the entries in its route reply packet table. A match indicates that this packet has previously been processed, and the packet is dropped. If there is no match, the node compares the uid (i.e., the trapdoor) in the packet against the entries in its routing data table. If there is no match, the node forwards the route reply after decrementing its TTL value. The node also adds the tuple:

(route_reply_token, uid, $\{m_E\}\text{nonce}_A$,timestamp, time_until_rebroadcast, route_nonce_list)

to its reply packet table. Each time any intermediate node re-broadcasts a route reply packet, it also broadcasts a RREPACK packet of the form:

[RREPACK, TTL = 1, route_reply_token].

Any node that receives this RREPACK packet searches its route reply packet table, and if it finds an entry with this route reply token, then the corresponding route reply is no longer forwarded.

3.3.8 Route Reply Packet processing at the Destination (the original source)

Upon receiving a route reply, the original source finds a match between the reply's uid and a uid stored in its routing data table. The source uses nonce_S and init_vector to decrypt $\{m_D\}\text{nonce}_S$. The source recovers ID_S in the decrypted clear text and updates the routing data table entry to include nonce_D and node_pseudonym_list. Before sending a data packet, the source calculates the session key by generating the hash of (ID_S , ID_D , nonce_S , nonce_D).

3.3.9 Data Transmission

The source generates a data packet with the format:

[DATA, TTL, data_token, uid, $\{\text{Data}\}K_{\text{session}}$, node_pseudonym_list]

The contents of the node_pseudonym_list are copied from arrived route reply. The uid serves as a trapdoor for the data packet. Before sending the data packet, the source stores the tuple:

(data_token, uid, $\{\text{Data}\}K_{\text{session}}$,timestamp,

time_until_rebroadcast rebroadcast_disabled, node pseudonym_list)

in its data packet table. The source continues to re-broadcast this packet until it receives a DATAACK (i.e., data acknowledgment) from any neighbor. Rebroadcast terminates when a DATAACK is received.

3.3.10 Data Packet Processing at an Intermediate Node

A node receiving a data packet first checks its list of pseudonyms for any element of the arrived data packet's node_pseudonym_list. If no match exists, the packet is dropped. If there is a match, this node processes the packet only if it hasn't previously seen the token. The node uses the trapdoor to determine if the packet is intended for this node. If unsuccessful, the node stores the tuple:

(data_token, uid, $\{\text{Data}\}K_{\text{session}}$, timestamp, time_until_rebroadcast, rebroadcast_disabled , node_pseudonym_list)

in its data packet table. The node decrements the packet's TTL and re-broadcasts it. A forwarding node also broadcasts a DATAACK other nodes use to disable rebroadcasts of the data packet.

3.3.11 Data Packet Processing at the Destination

The destination performs the same tests as an intermediate node. In this case, however the data packet's uid matches a uid value in the node's routing data table. The node uses the nonce and identity value corresponding to the matched uid to recover the symmetric session key and decrypt the data.

3.3.12 Route Error

Each time a node re-broadcasts a data packet, it waits for the corresponding DATAACK. If it does not receive a DATAACK within a pre-defined time period, the node sends a route error with the format:

[RERR, data_token, node_pseudonym_list].

Each node that receives a route error checks if it has previously processed the packet, and if not, propagates it. When the route error reaches the source, the source recognizes that the route used is invalid and deletes it from the route data table. It then initiates a fresh route discovery to the same node.

4 Performance Analysis

We implemented CARP in the *ns-2* network simulator. Cryptographic protocols are implemented based on existing available libraries (e.g., ibe-0.7.2 [http://crypto.stanford.edu/ibe/download.html]). The remaining cryptographic routines, such Blowfish [16] were implemented on existing OpenSSL libraries. Our simulation parameters are summarized in Table 4.

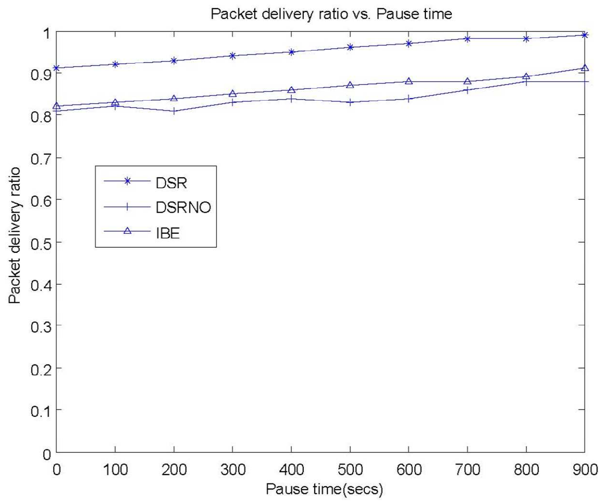
Protocols simulated	CARP, DSR, DSR-NO
Total number of nodes	50
Total number of sources	10
Maximum node velocity	20 m/s
Grid size	1500 x 300 m
CBR Rate	2 packets per second
Total simulation time	900 seconds
Node movement model	Random waypoint [18]
Total number of scenario files	20
Number of traffic files for each protocol	4

Table 4: Simulation Parameters

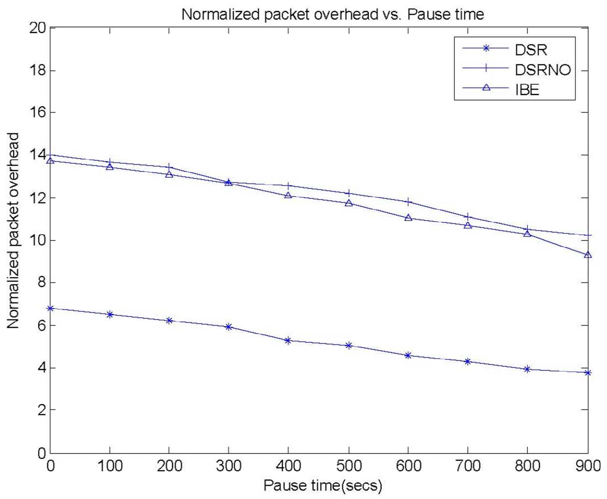
We compare our protocol with Dynamic Source Routing (DSR) [17], as well as DSR with all optimizations switched off (DSR-NO). As our algorithm does not include these optimizations, DSR-NO allows us to gauge the degree of performance degradation that is inherent in our protocol. In addition, the types of optimizations possible for our protocol are very different from DSR and are a subject for future work. The performance metrics we measured for each protocol are shown in Table 5.

packet delivery ratio	fraction of data packets sent that are received at the intended destination node.
normalized packet overhead	number routing control packets per data packet received; a route request sent over two hops counts as two packets
normalized byte overhead	number of routing control bytes per data packet received.
data latency	average time for data packet delivery

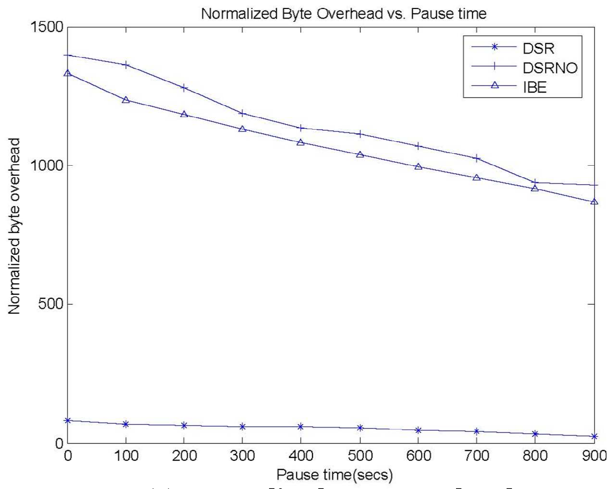
Table 5: Performance Metrics



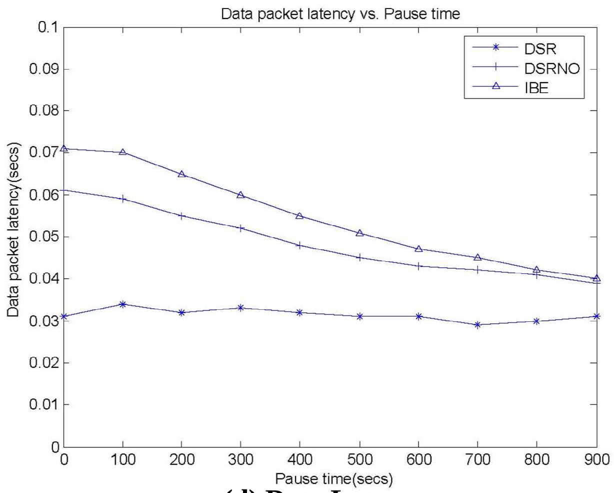
(a) Packet Delivery Ratio



(b) Normalized Packet Overhead



(c) Normalized Byte Overhead



(d) Data Latency

Figure 2: Performance results for 10 simultaneous sources

Figure 2 shows our performance comparison. In the graphs, "IBE" refers to CARP which is founded on Identity Based Encryption. Figure 2(a) shows the packet delivery ratios. Optimized DSR outperforms both CARP and DSR-NO as expected. The fact that a CARP node must perform some tests and/or cryptographic operations at every hop reduces the packet delivery ratio. As the pause times increase, the routes become more stable, and the packet delivery ratio increases. Figure 2(b) shows the normalized packet overhead for each protocol. CARP marginally outperforms DSRNO but within the margin of error. However, the figure shows that the *anonymity functionality* of CARP does not introduce additional overhead packets to the protocol. The difference between DSR and CARP results from DSR optimizations not implemented for CARP. Figure 2(c) shows a similar result. The results in this figure show that the protocol itself does not add overhead in terms of the size of the control packets. In comparison to approaches that require control packets to carry certificates [18], this shows a significant advantage of CARP. Figure 2(d) shows the average data latency of the protocols. The latency added by CARP results from increased processing (both simple comparisons and cryptographic operations) that CARP nodes perform as part of the protocol. The results are comforting, however, because added latency incurred to achieve significant anonymity should not be crippling to an application.

5 Security Analysis

CARP aims to provide identity, routing and location anonymity, along with complete data privacy. In an ad hoc network, with its lack of physical links as in a wired counterpart, messages are *untraceable*. This untraceability property can be exploited, as in the case of CARP, to provide strong anonymity properties to communication in a wireless network. This section examines in detail how immune the anonymity properties provided by CARP are, to various forms of attack that may occur in a mobile ad hoc network.

A fundamental tradeoff between CARP and some prominent secure routing protocols [20, 21, 22, 23, 24] is that the latter provide extensive node authentication which can only work in the presence of corresponding key management infrastructure. Our protocol is designed for strong anonymity with no node authentication or attendant key management infrastructure, making it very lightweight and robust. While the node authentication property allows such protocols to foil a number of denial of service attacks, unlike us they do not provide any anonymity guarantees. One could argue for anonymous credential schemes [31] but such credentials are short lived and expensive to create and maintain. So, although our protocol does not foil a number of denial of service attacks as effectively as the competing ones, under no circumstances do the anonymity or privacy guarantees fail. To understand these attacks, Table 6 depicts typical adversary models.

Observer	a passive adversary eavesdrops on all packets, without initiating malicious action; could function globally or locally
Disruptor	re-routes legitimate packets in dysfunctional ways, thereby disrupting normal route discovery and data transmission
Hostile Insider	deliberately initiates and breaks existing routes.
Compromised Insider	benign alone, but under control of a malicious node can enable access to concealed information
Compromiser	gains control of one or more nodes and forces them to initiate malicious action against other nodes; a variation is Active VC which partitions the network into groups of benign nodes and attempts to control and gather concealed information from all packets moving between the groups.

Table 6: Common Adversary Models

The attacks that can occur in an ad hoc network can be divided on the basis of their effects: routing disruption and excessive resource consumption. Some example attacks are route request flooding attack, packet looping attack, and blackhole and grayhole attacks, which are examined in detail below.

5.1 Disrupter Attacks

Disrupters may initiate looping and route request flooding attacks. These are denial of service attacks such as described above and do not impact CARP's anonymity guarantees. By design, each packet carries an unique token to prevent broadcast explosion and indirectly looping. Consider two malicious nodes that are in collusion to always reset the TTL value of each route request and re-broadcast it, to try to initiate a looping attack. If this re-broadcast packet now is intercepted by a benign intermediate node, it might get dropped because:

- an element in the pseudonym list might match an element in the receiving node's pseudonym list
- the packet's route_request_token might match an element in the node's list of seen tokens.

Even if this node were to re-broadcast this packet, it would decrement the packet's TTL, thereby breaking the looping condition.

A slightly complicated situation is when two nodes collude to reset the TTL *and* route_request_token (to a mutually agreed common value), of each route request they intercept. If the tampered route request packet reaches a node it may be incorrectly regarded as a fresh route request. Each destination, however only supports a fixed maximum number of routes, thereby preventing route replies from flooding the network. In such an attack, the concealed identity information in the route request will not be revealed.

A route request flooding attack is another disrupter attack that potentially bring down the network. A slight protocol modification would curtail this problem, which involves a time window during which a node would accept route requests, alternated with a period of time when it would not. Also, currently each destination only responds to a fixed number of route requests from a given source, preventing the network floods with unwanted route replies.

5.2 Hostile Insider Attacks

Wormhole and grayhole attacks are examples of hostile insider attacks. In the former, a malicious intermediate node drops all packets, and in the latter, the malicious node drops packets selectively. As mentioned previously, as our protocol does not provide for node authentication or any key management infrastructure, and therefore does not provide suitable means to foil this type of attack. However, it is guaranteed that none of the anonymity or secrecy requirements are violated.

5.3 Compromiser Attacks

A compromiser attack occurs when a malicious node compromises a benign node and uses it to gather identity and location information about the nodes the compromised node communicates with. To actually compromise a node, a malicious node has to launch a brute-force scanning attack to try to guess the node's identity, or gain the node's identity in some other fashion. In the brute-force approach, the malicious node tries all possible identities from the given identity space and for each guess, sends a route request packet, using a fake identity for itself. As per the protocol, the benign node would respond with a route reply. As the route reply packet would contain the benign node's actual identity and the malicious node's fake identity in encrypted form, the benign node's identity would be revealed. Even so, the malicious node must still somehow force the compromised node to send its routing data. In a dynamic mobile ad hoc network, it is also unlikely that the two nodes will remain connected long enough for this lengthy and expensive series of operations.

5.4 Tagging Attacks

We divide the tagging attacks into two categories: type A and type B. A type A attack is initiated by a malicious node with global information and launched against a pair of communicating nodes. The global malicious node could attach a specific node pseudonym value to a route request packet and track the packet through the network. It would not succeed in extracting the source or destination identities (as all identity information in route requests or replies are encrypted, with keys known only to the end parties), and at most it could determine the approximate area in which the destination is located. When the destination accepts the route request, the malicious node can identify the destination's approximate location but not its identity. While a malicious node can gain some information in this manner, the nature of mobile ad hoc networks radically decreases the likelihood that any one node ever has global knowledge of the network and its traffic.

In the type B attack, two nodes in collusion tag the packets by adding a specific pseudonym known to each other to the node_pseudonym_list of a route request packet. The nodes then monitor the route reply as it passes through the same two colluding nodes. The nodes could similarly monitor data packets. Although these two nodes could acquire some approximate information about the locations of the source and destination, they would not know their identities, or those of the other nodes in the route.

6 Related Work

Secure routing in ad hoc networks is an active research area. Prominent contributions include [18, 20, 21, 22, 23, 24, 25, 30]. Broadly, these can be classified on the basis of the cryptographic encryption-decryption schemes they use: symmetric [20, 21, 22, 23, 24] versus an asymmetric [18, 25,30] key based cryptographic approach. These approaches involve node authentication with attendant key management schemes, are often highly structured and hierarchical (as hash trees and chains) [23], which are difficult to establish and maintain in high mobility environments. In contrast, CARP does not require any node authentication or key management scheme to provide its strong anonymity guarantees. Additionally, the asymmetric key based schemes require public key certificate repositories with attendant maintenance costs. Thus, the ability of these protocols to successively foil most denial of service attacks, comes at increased computational cost, absent for CARP, which, with bare minimum use of asymmetric key cryptography, no authentication, minimal key management, and no public key certificate repositories, is truly a lightweight protocol for anonymous communication.

Anonymity in wired environments has been studied extensively [2, 3, 4], and researchers have started applying it to the mobile ad hoc environment [5, 26, 27]. These approaches adapt Mix-net [2] to the mobile ad hoc network. In particular, Hong. et al [27] propose using a cryptographic trapdoor, as is used in CARP, that allows intermediate nodes to decide whether or not to perform decryption operations when a *boomerang onion* arrives at a node. They however make extensive use of both asymmetric and symmetric key cryptographic schemes and signatures. This combination requires extensive key management infrastructure, which CARP avoids completely. In a peer-to-peer approach Zhao et al [5] uses a *probability of forwarding* to allow the nodes to decide which mix to send the next message, but encrypt the messages using the mix's public key, which implies the use of a key management infrastructure, absent in CARP. Moreover, no mention is made of how these cryptographic keys are distributed. The approach detailed by Deng et al [26] assumes that symmetric keys pre-exist between the nodes that want to communicate anonymously, implies the presence of a key management infrastructure, but does not explain key distribution.

The use of IBE [1] in mobile ad hoc networks has received a lot of attention recently [28, 29]. In fact, Arbaugh et al [28] propose a scheme that combines IBE with threshold cryptographic scheme to create servers that give IBE private keys to nodes, but the presentation details no implementation. Moreover, any threshold scheme involves increased messaging overhead during execution of the threshold scheme, as well as authentication of the node that wishes to obtain its private key, which in turn involves key certificates; CARP avoids both of these computational burdens. Agrawal et al [29] use a similar scheme, with the nodes participating in the threshold scheme generating both the IBE master secret as well as private keys of the nodes. Also, the IBE based authentication scheme is really a combination of digital signature and encryption, both of which are computationally expensive. CARP, in contrast limits use of IBE exclusively to the initial route request phase, thereby completely avoiding these computationally expensive operations.

7 Conclusion

Our anonymous routing protocol for ad hoc wireless networks provides strong anonymity guarantees with three major advantages, over competing schemes. First, CARP exploits the strong security guarantees of an asymmetric key cryptography scheme IBE, but eliminates the use of public key certificates and certificate management. By confining the use of IBE to the route request phase, CARP minimizes the costs of using such a scheme, and demonstrates that asymmetric key cryptographic schemes are practicable for wireless ad hoc networks, if used judiciously. Moreover, CARP exploits the lack of links in an ad hoc network to provide strong anonymity guarantees, such as identity, routing and location identity anonymity for nodes on an ad hoc network. Simulation results obtained with the popular ns-simulator are in good agreement with expected outcome, and demonstrate that this protocol can be applied to real-world ad hoc networks.

Bibliography

[1] D. Boneh and M. Franklin. Identity Based Encryption From Weil Pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.

[2] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

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

[4] R Dingledine, N Mathewson, and P Syverson. Tor: The Second Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*, pages 303–320, August 2004.

[5] S. Jiang, N.H. Vaidya, and W. Zhao. A Mix-net algorithm for Mix-net in Wireless Ad Hoc Networks. In *Proceedings of the 2004 International Conference on Mobile Ad-hoc and Sensor Systems*, pages 406–415, October 2004.

[6] C. Perkins and P. Bhagwat. Highly Dynamic Destination -Sequenced Distance-Vector Routing (DSDV) for Mobile Computers, In *Proceedings of ACM SIGCOMM 94,* pages 234–244 1994

[7] S. Murthy and J.J Garcia-Luna Aceves. A Routing Protocol for Packet Radio Networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking, MobiCom'95*, pages 86–95, November 1995

[8] D. Johnson, D. Maltz, and J. Broch. DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks. In Ad Hoc Networking, edited by Charles E. Perkins, Chapter 5, pages 139–172, Addison-Wesley, 2001.

[9] C. Perkins and E. Royer. Ad-Hoc On Demand Distance Vector Routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing System and Applications*, pages 90-100, February 1999.

[10] D. Liu, P. Ning and R. Li. Establishing pairwise keys in distributed sensor networks. *ACM Transactions on Information and System Security,* 8(1):41–47, February 2005.

[11] BJ Kwak, NO Song, LE Miller. Analysis of the Stability and Performance of Exponential Backoff In *Proceedings of IEEE WCNC*, pages 1754 -1761, 2003.

[12] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu. The Broadcast Storm Problem in a Mobile Ad Hoc Networks. Wireless Networks 8, pages 153 -167, 2002.

[13] Kong, X Hong, M Gerla.n ANODR ANonymous On Demand Routing With Untraceable Routes for Mobile Ad Hoc Networks. In *Proceedings of MobiHoc 2003,* pages 291 – 302, June 2003.

[14] N. Koblitz, A. Menezes and S. Vanstone. The State of Elliptic Curve Cryptography. *Designs, Codes and Cryptography*. 19(2-3):173–193, March 2000.

[15] V. Miller. The Weil Pairing and its Efficient Calculation. *Journal of Cryptology*, 17(4):235–261 September 2004.

[16] B. Schneier. Description of a New Variable-Length Key 64-bit Block Cipher (Blowfish). In Proceedings *of Cambridge Security Workshop,* pages 191–204, December 1993.

[17] D. Johnson, D. Maltz, and J. Broch. DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks. In *C. Perkins, editor, Ad Hoc Networking,* chapter 5, pages 139--172. Addison-Wesley, 2001.

[18] K Sanzgiri, B Dahill, BN Levine, C Shields, E. M Royer. A Secure Routing Protocol for Ad Hoc Networks In *Proceedings of the 10th IEEE International Conference on Network Protocols* November 2002

[19] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of Mobile Computing and Networking (MobiCom 1998),* pages. 85–97, 1998.

[20] Y-C Hu, A. Perrig, and D. B. Johnson. Rushing Attacks and Defence in Wireless Ad Hoc Network Routing Protocols In *Proceedings of the 2003 ACM Workshop on Wireless Security (WiSe 2003),* pages 30–40, September 2003.

[21] Y-C Hu, A. Perrig, and D. B. Johnson. SEAD Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. *Ad Hoc Networks* 1(1):175–192, July 2003.

[22] Y.-C. Hu, A. Perrig, and D. B. Johnson. Packet Leashes: A Defence Againt Wormhole Attacks in Wireless Ad Hoc Networks. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, pages 1976-1986, April 2003.

[23] 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)*, pages 57–73, February 2003.

[24] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne: A Secure Routing Protocol for Ad Hoc Networks. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking (MobiCom 2002)*, pages 12–23, September 2002.

[25] J.P. Hubaux, L Buttyan, S Capkun. The Quest for Security in Mobile Ad Hoc Networks. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, pages 146–156, 2001.

[26] B. Zhu, Z. Wan, M. Kankanhalli, F. Bao, R. Deng. Anonymous Secure Routing in Mobile Ad-Hoc Networks. In *Proceedings of the 29th International Conference on Local Computer Networks.* pages 102-108, 2004

[27] 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 and Ad Hoc Networking and Computing.* pages 291-302, June 2003

[28] A. Khalili, J. Katz, and W. Arbaugh. Toward secure key distribution in truly ad-hoc networks. In *Proceedings of IEEE Security and Assurance in Ad-Hoc Networks at International Symposium on Applications and the Internet* , pages 342-346, 2003.

[29] H.Deng, A.Mukherjee and D. Agrawal Threshold and Identity-based Key Management and Authentication for Wireless Ad Hoc Networks. In *Proceedings of IEEE International Conference on Information Technology*, Volume 1 pages 107-112, 2004

[30] L. Zhou and Z. Haas. Securing ad hoc networks IEEE Network Magazine Special Issue on Network Security, vol. 13, no.6, Nov-Dec. 1999

[31] J. Camenisch and A. Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. B.Pfitzmann (Ed.) EUROCRYPT 2001 LNCS 2045 pages 93 -118, 2001 Springer-Verlag.