A Novel Solution for Achieving Anonymity in Wireless Ad hoc Networks[1]

Azzedine Boukerche[†], Khalil El-Khatib[‡†], Li Xu[†], Larry Korba[‡]

[†]PARADISE Research Laboratory
SITE, University of Ottawa, Canada
{boukerch,lxu}@site.uottawa.ca

[‡]National Research Council Canada
Ottawa, Canada
{khalil.el-khatib,larry.korba}@nrc.gc.ca

ABSTRACT
A mobile ad hoc network consists of mobile nodes that can move freely in an open environment. Communicating nodes in a wireless and mobile ad hoc network usually seek the help of other intermediate nodes to establish communication channels. In such an open environment, malicious intermediate nodes can be a threat to the security and/or anonymity of the exchanged data between the mobile nodes. While data encryption can protect the content exchanged between nodes, routing information may reveal valuable information about end users and their relationships. The main purposes of this paper are to study the possibility of achieving anonymity in ad hoc networks, and propose an anonymous routing protocol, similar to onion routing concept used in wired networks. Our protocol includes a mechanism to establish a trust among mobile nodes while avoiding untrustworthy nodes during the route discovery process. The major objective of our protocol is to allow only trustworthy intermediate nodes to participate in the routing protocol without jeopardizing the anonymity of the communicating nodes. We present our scheme, and report on its performance using an extensive set of simulation set of experiments using ns-2 simulator. Our results indicate clearly that anonymity can be achieved in mobile ad hoc networks, and the additional overhead of our scheme to DSR is reasonably low when compared to a non-secure DSR ad hoc routing protocol.

Categories and Subject Descriptors
C.2 [**Computer-Communication Networks**]: Distributed Systems; C.4 [**Performance Systems**]: Modeling Techniques;

General Terms
Algorithms, Performance, Security

[1] This work was supported by the Canada Research Chair program, NSERC, Canada Foundation for Innovation Funds, and OIT/Distinguished Researcher Award

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
PE-WASUN'04, October 7, 2004, Venizia, Italy.
Copyright 2004 ACM 1-58113-959-4/04/0010…$5.00.

Keywords
Routing, Ad hoc, Security, Network Simulator *ns*-2, Wireless Networks

1. INTRODUCTION
Recent advances in wireless and mobile communication technologies coupled with the recent proliferation of portable computer devices have led the development efforts for future wireless networks towards wireless and mobile ad hoc networks. The attention that this type of networks has received is mainly due to their potential applications in commercial and military applications.

In an ad hoc network, two nodes can communicate directly as long as they are within the radio communication range of each other, but when the two nodes are far apart, they require the help of other intermediate nodes to relay their traffic. However, since there is no stationary infrastructure such as network routers, all network nodes have to cooperate in relaying each other data. But in such hostile environments, the information exchanged between two communicating parties might include highly sensitive data that must be secured when sent through intermediate nodes. While end-to-end security mechanisms can provide some level of security for the data, valuable information, such as location and relationships of the communicating entities may easily be determined from traffic and data analysis. Network-based anonymity techniques, for instance, may offer the prospect of hiding this information.

For the Internet, several network-based anonymity approaches provide anonymous communication between end-nodes. These approaches include DC-nets [7], Crowds [20], MIX networks [6], and Onion Routing [19]. Both MIX networks and Onion Routing share the same concept of establishing anonymous paths for the *data transfer*. To construct an anonymous path, a source node must store and maintain information about the topology of the network. But keeping up-to-date information about the topology of the network is complex in the absence of fixed infrastructure and in the presence of dynamic topology, as is the case with ad hoc wireless networks.

In this paper, we present a novel secure distributed path construction protocol for anonymous communication in wireless ad hoc networks. As opposed to other related protocols, our protocol does not require the source node to gather and store

information about the network topology. Instead, the source node initiates a path establishment process by broadcasting a *path discovery* message with certain trust requirements to all of neighboring nodes. Intermediate nodes satisfying these trust requirements insert their identification (IDs) and a session key into the *path discovery* message and forward copies of this message to their selected neighbors until the message gets to its destination. The intermediate nodes encrypt this information before adding it to the message. Once the receiver node receives the message, it retrieves from the message the information about all intermediate nodes, encapsulates this information in a multi-layered message, and sends it along a reverse path in the dissemination tree back to the source node. Each intermediate node along the reverse path removes one encrypted layer from the message, and forwards the message to its ancestor node until the message reaches the source node. When the protocol terminates, the source node ends-up with information about all the trusted intermediate nodes on the discovered route as well as the session keys to encrypt the data transmitted through each of these nodes. The multicast mechanism and the layered encryption used in the protocol ensure the anonymity of the sender and receiver nodes. In this paper, we will give a general review of the protocol and present some of the early performance result of the protocol.

The remainder of the paper is organized as follows. Section 2 discusses the anonymity and security issues in wireless ad hoc networks. Section 3 describes briefly the trust management system upon which our algorithm relies. Section 4 introduces our secure distributed anonymous routing protocol, which we refer to as SDAR. Section 5 reports on the simulation-based performance analysis of our scheme. Section 6 concludes the paper.

2. ANONYMITY AND SECURITY IN MOBILE AD HOC NETWORKS

A variety of widely known intrusion techniques may be used to infer the entities' identities, their locations, and/or relationships between communicating entities in a public network. Typical malicious actions may affect the message coding, timing, message volume, flooding, intersection and collusion. Onion Routing [19] is a communication protocol that is resistant against some of these attacks. It employs a network of Chaum MIXes [6] in order to provide anonymous and secure communications. It provides a communication infrastructure that is reasonably resilient against both eavesdropping and traffic analysis. Using this protocol, entities representing applications communicate through a sequence of networked computing nodes, which is referred to as onion routers. Onion routers are generally application layer routers that realize Chaum MIXes. Onion routing connections proceed in three phases: connection setup phase, data transfer phase and connection termination phase.

Over the Internet, anonymous systems [8], [10], [23] use application level routing to provide anonymity through a fixed core set of MIXes, as we described earlier for the Onion Routing protocol. Each host keeps a global view of the network topology, and make anonymous connections through a sequence of MIXes instead of making direct socket connections to other hosts. The authors in [13] used an alternate Onion Routing approach to provide anonymous communications for mobile agents in the JADE environment (Java Adaptive Dynamic Environment). Each JADE multi-agent has several onion agents that provide an anonymous data forwarding service, and at least one onion monitor agent that keeps track of the location of all other onion agents in the system. Onion monitor agents exchange onion agent reachability information in order to maintain a valid topology of the complete onion agent network. Levien [1], [15] developed a monitoring utility that queries MIXes and publishes on a website the average latency and uptime of each MIX over the past 12 days. Recently, Tarzan [9] and MorphMix [21] have discussed the difficulties of constructing routes in dynamic environments.

Achieving secure routing in wireless ad hoc networks is a complex task due to the nature of the wireless environment and the lack of predefined infrastructure [14]. A number of protocols have been developed to add security to routing in ad hoc networks. Papadimitriou and Haas [17] proposed SRP (Secure Routing Protocol) based on DSR [11], [12]. The protocol assumes the existence of a security association between the source and destination to validate the integrity of a discovered route. Sanzgiri *et. al.* [22] proposed the ARAN (Authenticated Routing for Ad hoc Networks) protocol that uses public key cryptography instead of the shared security association used in the SRP [17]. Each intermediate node running the protocol verifies the integrity of the received message before forwarding it to its neighbor nodes. Source and destination nodes use certificates included in the route discovery and reply messages to authenticate each other. The protocol has an optional second discovery stage that provides non-repudiating route discovery. Yi [25] developed a generalized SAR (Security-Aware Ad-hoc Routing) protocol for discovering routes that meet a certain security criteria. The protocol requires that all nodes that meet a certain criteria share a common secret key.

Venkatraman and Agrawal [24] proposed an approach for enhancing the security of AODV protocol [18] based on public key cryptography. In their approach, two systems, EAPS (External Attack Prevention System) and IADCS (Internal Attack Detection and Correction System) were introduced. EAPS works under the assumption of having mutual trust among network nodes while IADC runs by having the mutual suspicion between network nodes. Every route request message carries its own digest encrypted with the sender's private key hash result in order to ensure its integrity. To validate established routes, route replies are authenticated between two neighbors along them. This approach prevents external attacks. IADC system classifies internal attacks and sets a misbehavior threshold for each class of attack in order to detect compromised network nodes.

The above three protocols, i.e., SRP, ARAN, and Venkatraman and Agrawal's schemes, ensure only the authenticity but not the privacy of the routing information, while SAR finds routes that meet a certain security level. In all these protocols, intermediate nodes that handle the route control messages can easily find the identity of the communicating nodes, which must be protected in case of anonymous communication. Our protocol uses the Onion Routing approach and trust management system to provide trust and anonymity for the path discovery (and hence for subsequent communications using this path).

Other studies include intrusion detections to study the behavior of mobile users in mobile phone systems [3], [4].

3. TRUST MANAGEMENT SYSTEM

As we mentioned earlier, due to the openness of ad hoc wireless environment, some nodes in the network are likely to defect and become harmful to the network, thereby necessitating a mechanism to identify these nodes and isolate them. In this section, we will introduce the notion of trust management system we have used in our proposed protocol. The purpose of this system is to motivate the participating nodes not only to help each other relaying data traffic, but also identify the malicious nodes, and avoid using them during the route establishment. The identification of malicious nodes makes it easy to take them out of the network, thereby increasing the route's security and reliability

In this section, we will introduce our trust management approach as well as the trust notion we choose to use in ad hoc wireless environment to select routing path that meets certain trust requirements. In our approach, we define the trust level in a node as a cumulative value that is based on the past behavior of the node. The trust level of a node increases as long as the node behaves exactly as it is supposed to (in our cases, follow reliably the steps of the routing protocol) or decreases as the node misbehaves accordingly. A node's trust is computed by each of its direct neighboring nodes based on their past experience or observation of the node's behavior. These neighboring nodes, together with the evaluated node, form what we refer to as a community, as we will describe later.

3.1 Community management

In our system, we define a node's community as the set of nodes that includes the node itself, referred as central node, and all of its one-hop neighboring nodes, among which some may be malicious. To build and maintain a node's community, we employ a similar method used by AODV ad hoc routing protocol [18] in order to accomplish neighboring nodes management. In our protocol, a node keeps track of its neighbors simply by listening for a HELLO message, which is broadcasted periodically by each node. The sender's public key is passed as part of the HELLO message. Upon receipt of a HELLO message from one of its neighboring nodes, a central node stores its neighboring node's public key if it does not have it yet. Since nodes can move freely in an ad hoc wireless network, some neighbors of the central node may leave while new neighbors may join the neighborhood of the central node. Thus, if a node does not receive for some time the HELLO message from one of its neighbors, it removes it from its list identifying its neighboring nodes.

3.2 Community Key Management

In each *community*, the *central node* classifies its neighboring nodes into three classes, based on their trust level. The first and lowest trust level is for nodes whose trust value is between 0 and a *Medium Trust Level Threshold*, $\delta 1$, while the second trust level, i.e., the medium level, contains the nodes whose trust level is between $\delta 1$ and the *High Trust Level Threshold*, $\delta 2$. The trust level, corresponding to the high level, contains the nodes whose trust value is between $\delta 2$ and *1*. Each node selects independently the values for $\delta 1$ *and* $\delta 2$. In our experiments, both values have been determined empirically.

The c*entral node* generates two different keys for the medium and high trust level, and shares them with its neighbors. All neighbors in the same trust level share the same key. The neighbors in high trust level will have both High Trust Level Community Key (referred to as HCK) and Medium Trust Level Community Key (referred to as MCK); whereas, the neighbors in medium trust level have only MCK. As for the neighbors in low trust level, they do not share any community key at all.

When the central node detects a new neighbor, it will assign an initial trust value to it and updates this trust level later on, based on their interaction. The central node updates the corresponding community key when a node's trust level goes up or down, and also when a node leaves the community. To protect a community key during distribution, the central node encrypts the key with the public key of the intended neighboring node before sending it. The central node repeats the same process with each node in its neighboring node set.

3.3 Identification of Nodes' Malicious Behavior

In this section, we will describe how each node can compute and constantly update the node's trust in its neighboring nodes. Our approach is based on the ability of the node to identify neighboring nodes good or malicious behavior, and hence updating the trust level accordingly. A behavior is good if it confirms to the specification of the routing protocol and malicious otherwise. For our protocol, a malicious behavior happens when a node drops silently the packet without forwarding it or maliciously updating the packet before forwarding it. We call these two malicious behaviors as *Malicious Dropping* and *Malicious Modification*. A node can identify these behaviors simply by overhearing whether its neighboring node modified maliciously the message before sending it (*Malicious Modification*) or simply did not forward the message (*Malicious Dropping*). Note that for the destination node to protect its anonymity without jeopardizing its trust, it must also forward a copy of the message it receives.

3.4 Trust-Based Distributed Route Selection Scheme

Our routing protocol, as we shall see in the next section, requires each intermediate node that receives a *route request* message, to forward this message to its neighboring nodes. But in order to achieve the security and reliability of the route, our protocol uses a selection algorithm that is based on the level of trust each intermediate node has with its neighboring nodes.

When a source node initiates the route discovery protocol, it specifies the trust level requirement in the initial message. Each intermediate node will propagate the message only to selected neighboring nodes, depending on the source node requested trust level. If the requested trust level is *high*, the node will use the community key for the neighbors with high trust level to encrypt the message; this will ensure that only highly trusted nodes will participate in the routing protocol. If the required trust level is *medium*, the node will use the community key for the neighbors with medium or high trust level to encrypt the message. Using this approach restricts the participation of intermediate nodes only to the ones that have a certain trust level.

4. SECURE DISTRIBUTED ANONYMOUS ROUTING PROTOCOL (SDAR)

In this section, we will describe our secure distributed protocol for establishing anonymous paths in ad hoc wireless networks. The major objective of our protocol is to allow trustworthy intermediate nodes to participate in the path construction protocol without jeopardizing the anonymity of the communicating nodes. The protocol has a number of characteristics, including Non-Source-Based Routing, Flexible and Reliable Route Selection and Resilience against Path Hijacking. Our protocol is secured against passive and active attacks, but not against Denial-of-Service attacks, it maintains the anonymity of the sender and receiver, and it is able to establish a route matching certain trust level requirement if enough nodes with qualifying trust value exist between the source and destination. The interested reader may wish to consult [2].

4.1 Overview

To send data anonymously to a receiver node *R*, a sender node *S* has to discover and establish a reliable and anonymous path that connects the two nodes. Both the *path discovery* and establishment process should be carried out securely and without jeopardizing the anonymity of the communicating nodes. The process is divided into three phases: the *path discovery* phase, the *path reverse* phase and the *data transfer* phase. Distributed information gathering about intermediate nodes that can be used along an anonymous path is carried out during the *path discovery* phase, while passing this information to the source node takes place during the *path reverse* phase. The official data exchange is processed during the *data transfer* phase after the construction of the route. We elaborate on these three phases during the following sub-sections, but we first introduce the assumption and some main definitions that are used henceforth.

Table 1: Notations

• ID_i:	The identity of node *i*.
• PK_i:	The public key of node *i*.
• *TPK*:	A temporary one-time public key.
• *TSK*:	The private (secret) key corresponding to *TPK*.
• K_i:	A symmetric (session) key generated by node *i*.
• PL_S:	The padding length set by the sender.
• P_S:	A padding implemented by the sender.
• PL_R:	The padding length made by the receiver *R*.
• P_R:	A padding made by the receiver node *R*.
• $E_{PK_i}(M)$:	The *message M* is encrypted with a public key PK_i.
• $E_{K_i}(M)$:	The message *M* is encrypted with the symmetric session key K_i.
• *H (M)*:	The message *M* is hashed with a hash function.
• $H_{K_i}(M)$:	The mixture of M and K_i is hashed with a hash function.
• $Sign_S(M)$:	The message *M* is signed with the private key of the source node *S*.
• $SN_{session\_ID_i}$:	A random number generated by node ID_i for the current session.
• HCK_i:	The high trust level community key which is a one way symmetric key and generated by node *i*.
• MCK_i:	The medium trust level community key which is a one way symmetric key and generated by node *i*.

4.2 Assumptions and Definitions

Before we proceed further, we will make the following assumptions about the ad hoc network.

- The links between wireless nodes are always bi-directional.
- Every wireless node has enough computation power to execute encryption and decryption algorithm.
- There is a trusted certificate authority (CA) outside the ad hoc network, which issues public key and private key to the wireless nodes inside the network.
- Each wireless node holds only one IP address for its communication in the ad hoc network, by which it will be recognized by all other wireless nodes.
- There are some nodes that are not willing to cooperate for routing and data delivering and possibly actively intent to tamper the routing protocol.

Table 1 shows the main notations used in this paper.

4.3 Path Discovery Phase

The *path discovery* phase allows a source node *S* that wants to communicate securely and privately with node *R* to discover and establish a routing path through a number of intermediate wireless nodes. An important characteristic of this phase is that none of the intermediate nodes that participated in the *path discovery* phase can discover the identity of the sending node *S* and the receiving node *R*.

The source node *S* triggers the *path discovery* phase by sending a *path discovery* message to all nodes within its wireless transmission range. The *path discovery* message has five parts. The first part is the open part. It consists of message type, *TYPE*, trust requirement, *TRUST_REQ*, and a one-time public key, *TPK*. The trust requirement indicated by *TRUST_REQ* could be *HIGH, MEDIUM* or *LOW*. *TPK* is generated for each *path discovery* session and used by each intermediate node to encrypt routing information appended to the *path discovery* message. This key serves also as a unique identifier for the message. The second part contains the identifier ID_R of the intended receiver, the symmetric key K_S generated by the source node and PL_S the length of the third part, *padding*, all encrypted with the public key PK_R of the receiver. The source node may learn about the public key PK_R of the destined receiver through a number of ways including using the service of a certificate authority (CA). The symmetric key K_S is used to encrypt the fourth part of the message as well as to protect against *replay attacks*. The third part is a padding P_S, generated by the source node and used to hide real routing information and to protect against message size attack. The forth pare consists of ID_S, PK_S, *TPK*, *TSK*, $SN_{Session\_ID_S}$ and $Sign_S(M_S)$, all encrypted with K_S. The intended receiver uses the public key *TPK* and its corresponding private key *TSK* to decrypt and verify the routing information in the message. $SN_{Session\_ID_S}$ is a random number generated by the source node and is mapped to the encryption key K_S to use with the message. $Sign_S$ protects the integrity of the message. The fifth part of the message contains information about intermediate nodes prior to the current node along the route. A message just sent by a source node has the

format shown in Figure 1, with M_S = *H (TYPE, TRUST_REQ, TPK, TSK,* ID_R, K_S, ID_S, PK_S, $SN_{Session\_ID_S}$, PL_S, P_S*).*

$$TYPE, TRUST\_REQ, TPK,$$
$$E_{PK_R}(ID_R, K_S, PL_S),$$
$$P_S,$$
$$E_{K_S}(ID_S, PK_S, TPK, TSK, SN_{Session\_ID_S}, Sign_S(M_S))$$

Figure 1. Path discovery message just sent by the source *S*.

We assume that each node keeps an internal table for mapping the randomly generated number of a session to the encryption key for the session, as well as to the ancestor and successor node along the anonymous path for the session. Given an encrypted message and a randomly generated number, a node can use this mapping table to know which key to use to encrypt the message. Only the random number, the session key, and the ancestor node entry are added to the table during the path discovery phase, while the successor node entry is added later during the path reverse phase.

When a node *i* receives a *path discovery* message, it processes the message according to the following steps:

1. Check if the message has already been received from other nodes within its wireless transmission range using the *TPK* as the unique identifier for the message. If the message was received previously, drop it silently and stop; otherwise, continue.
2. Check if the node is the sender's intended next hop by finding the corresponding community key in its community key lists. If the key is found then decrypt the message using that key and go to the next step; otherwise, stop
3. Check if the node is the destined receiver (try to decrypt $E_{PK_R}(ID_R, K_S, PL_S)$, with the private key of the node and compare the IDR to the node's id)
4. If the node is NOT the intended receiver, then
   a. Add the following information to the message, all encrypted with the *TPK*: the id of the node, a session key K_i (shared encryption key generated by the node), a randomly generated number $SN_{Path\_ID_i}$ for the session, and the signature of the original received message.
   b. Forward the new message to the neighbors whose trust levels meet the source node's trust requirement.
   c. Add $< SN_{Path\_ID_i}$, id of the ancestor node, $K_i >$ to the internal mapping table.
5. If the node is the destined receiver, then
   a. Use the length of padding, PL_S, from $E_{PK_R}(ID_R, K_S, PL_S)$ to find out the offset of the fourth part and then use the retrieved session key *KS* to decrypt the fourth part of the message and get *TSK*, then use the *TSK* to get session keys for all the nodes along the path of the message.
   b. Put all ids of the nodes and their session keys in one message; encrypt the message several times, each time with the session key of a node along the path to the receiver. Use the reverse order of the keys in the message (same as the data flow in onion routing)
   c. Send the message to the first node in the reverse path

A path discovery message that has already traveled nodes i on its way from the sender S to the receiver R would have the format shown in Figure 2, with *MS = H(TYPE, TRUST_REQ, TPK, TSK,* ID_R, K_S, ID_S, PK_S, $SN_{Session\_ID_S}$, PL_S, P_S*)*, and M_{ID_i} = H (M_{prev}, ID_i, K_i, $SN_{Path\_ID_i}$), and M_{prev} is the cumulative message that $node_i$ gets from its ancestor $node_{i-1}$.

$$TYPE, TRUST\_REQ, TPK,$$
$$E_{PK_R}(ID_R, K_S, PL_S),$$
$$P_S,$$
$$E_{K_S}(ID_S, PK_S, TPK, TSK, SN_{Session\_ID_S}, Sign_S(M_S)),$$
$$E_{TPK}(ID_1, K_1, SN_{Session\_ID_1}, Sign_{ID_1}(M_{ID_1})),$$
$$\vdots$$
$$\vdots$$
$$E_{TPK}(ID_i, K_i, SN_{Session\_ID_i}, Sign_{ID_i}(M_{ID_i}))$$

Figure 2. Path discovery message just processed by node$_i$.

4.4 Path Reverse Phase

The *path discovery* message is forwarded from one node to the other in the network until it reaches the target receiver *R*, which triggers the *path reverse* phase. When the intended receiver gets the *path discovery* message, it can use its private key to retrieve K_S. Then using K_S, it can obtain the temporary private (secret) key *TSK* encrypted in the fourth part of the message. Using *TSK*, the receiver node *R* can also retrieve the IDs of all intermediate nodes and the session key to use with each one of these intermediate nodes, and the random number generated by each node. The receiver then composes a message that contains all these random numbers and the corresponding session keys, and encrypts the message with the session keys of all the nodes along the path to the source node. With each encryption, the receiver *R* adds a layer that contains the random number generated by the node and the random number generated by the node's next-next-hop node along the reverse path to the sender. If the first node to get this message from the receiver is node *i*, the encrypted message constructed by the receiver *R* should have a format completely similar to the format shown in Figure 3, where $M_i = E_{K_{i-1}}(M_{i-2}), SN_{Session\_ID_i}, SN_{Session\_ID_{i-2}} H(M_{i-2}), H_{K_i}(N_i)$, N_i = ($E_{K_i}(M_{i-1}), SN_{Session\_ID_i}, SN_{Session\_ID_{i-2}}, H_{K_i}(N_i)$). P is a padding that has the same length as any M_j, and $SN_{Sesson\_ID_{S-1}}$ is a random number having the same number of bits as any regular $SN_{Sesson\_ID_j}$ and it is generated by the source node.

$$
\begin{aligned}
&TYPE,\\
&E_{K_i}(E_{K_{i-1}}(E_{K_{i-2}}...(E_{K_2}(E_{K_1}(E_{K_S}(\\
&SN_{Session\_ID_1}, K_1, SN_{Session\_ID_2},\\
&K_2,...,K_i, SN_{Session\_ID_R}, PL_R, P_R),\\
&SN_{Session\_ID_S}, SN_{Session\_ID_{S-1}}, H(P), H_{K_S}(N_S)),\\
&SN_{Session\_ID_1}, SN_{Session\_ID_{S-1}}, H(P), H_{K_1}(N_1)),\\
&SN_{Session\_ID_2}, SN_{Session\_ID_S}, H(M_S), H_{K_2}(N_2)),...),\\
&SN_{Session\_ID_{i-2}}, SN_{Session\_ID_{i-4}}, H(M_{i-4}), H_{K_{i-2}}(N_{i-2})),\\
&SN_{Session\_ID_{i-1}}, SN_{Session\_ID_{i-3}}, H(M_{i-3}), H_{K_{i-1}}(N_{i-1})),\\
&SN_{Session\_ID_i}, SN_{Session\_ID_{i-2}}, H(M_{i-2}), H_{k_i}(N_i)
\end{aligned}
$$

Figure 3. Path reverse Message

Each intermediate node that receives the *path reverse* message uses the $SN_{Session\_ID_i}$ to retrieve the key for the session, removes one encryption layer and forwards the message to the next node on the reverse path to the source node. The *ID* of the node from which the message was received is added to the successor node entry corresponding to the random number into the mapping table. When the source node receives the message, it decrypts the message and passes the information about all the intermediate nodes (i.e., the route) to the higher application.

4.5 Data Transfer Phase

Our protocol uses a similar approach to the Onion Routing protocol for the data transfer.

When the source node gets the *path reverse* message, it first checks whether or not the message is correct, and then uses the shared session keys of the intermediate nodes to make the layer encryption for the data, which the sender wants to transfer to the receiver. Each intermediate node just decrypts one encryption layer and forwards the message to the next node according to the *ID* of the next node.

5. SIMULATION EXPERIMENTS

In this section, we present the simulation experiments we have carried out to evaluate the performance of our protocol using the Network Simulator, *ns-2* [16]. During our experiments we wish to be able to study closely the behavior of our protocol. In this paper, we present the results we have obtained using an ad hoc network which consists of 30 mobile nodes, moving according to the *random waypoint* mobility model [5], and where the initial position of each mobile node was randomly chosen. Except for the experiments in Section 5.1 where we changed the speed value, the nodes moved around at a maximum speed of 5m/s in a randomly chosen direction within a flat rectangle area of 670m x670m. The nodes had to pause for a configured period of time, which we set it up to 20s, before they changed their moving direction. In each experiment, the simulation was run for 4000 sec of simulated time. Half of the route requests have High Trust Requirement (HTR) for the intermediate nodes and the other half have Low Trust Requirement (LTR). The values of the *Medium Trust Level Threshold*, *δ1*, and the *High Trust Threshold*, *δ2*, were empirically selected as *0.6* and *0.95*.

We have used an IEEE 802.11 MAC layer and a 914MHz Lucent WaveLAN DSSS radio interface with the transmission range of 250m. The traffic was generated by CBR sources over UDP. The source node continuously generated data packets of 512 bytes at the rate of 4 packets per second in each flow.

SDAR was simulated using 512 bit keys for authentication with the RSA algorithm, 64 bit for the community keys, and 64 bit session keys. We believe these values are reasonable given the computation resource of the mobile ad hoc nodes.

We have evaluated our protocol using the following performance metrics:

- **Connectivity:** Connectivity is defined as the percentage of the route request successfully answered by the destination, out of the route requests sent by the source.
- **Number of Packets:** In our simulations, we kept track of the number of packets that were prepared to be sent, the number of packets actually sent, and the number of packets received.
- **Routing Overhead**: Routing overhead represents the number of routing packets required to send one hundred data packets.
- **End-to-End Delay:** End-to-end delay is defined as the average time difference between the time a packet is sent from the source and the time it is successfully received by the destination.
- **Average Route Length:** the average route length indicates the average number of hops in a route.
- **Security Overhead:** The security overhead indicates the number of UPDATE messages sent by all nodes. These messages, as we mentioned in Section 3, are used to compute and update the trust value of mobile nodes.

5.1 SDAR and DSR- A Comparison

In this section, we wish to investigate the overhead of our scheme when compared to the best well known routing algorithm. In this paper, we choose the Dynamic Source Routing (DSR) routing protocol [11], [12]. This was due mainly to the fact that both protocols share the main routing idea. During the course of our simulations experiments, both protocols were run under identical mobility and traffic scenarios. We used a basic version of DSR, which does not include any optimization, such as the caching, which, as one may expect, goes against the anonymity concept. This allowed us, also, to have a fair comparative study.

Figure 4 and Figure 5 show that the SDAR protocol has a lower connectivity rate and a higher end-to-end delay. This variation is mainly due to the computational overhead of the security functions required for anonymously computing the path; intermediate nodes have to spend more time processing each single message, leading to a larger waiting time and hence longer end-to-end delay.

5.2 The effect of the change in the percentage of malicious node on SDAR

In the experiments we have described in the previous section, we have compared mainly the security overhead incurred by the SDAR, in comparison with the basic version of DSR [11], [12]. In

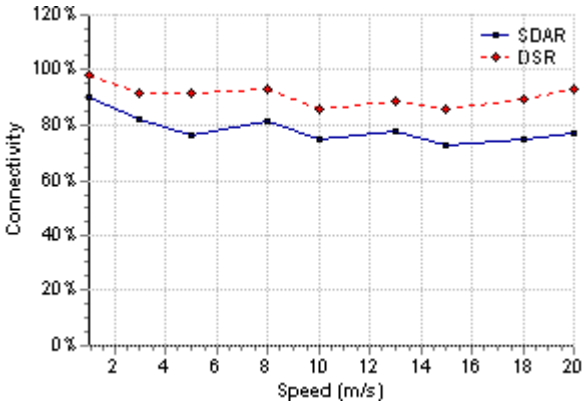

Figure 4. Connectivity of SDAR vs. DSR

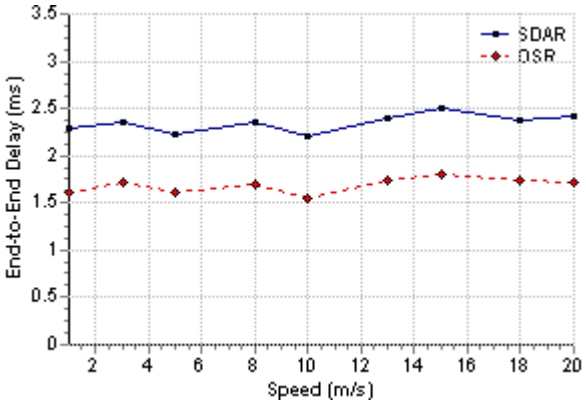

Figure 5. End-to-end delay for SDAR vs. DSR

these experiments, we have also assumed that all the nodes in the network are well-behaved nodes. In this section, we will present the additional experiments we have conducted to determine the effect of the change in the percentage of malicious nodes on the behavior of the SDAR protocol.

Figure 6 shows the change in the percentage of connectivity in relation with the change in the percentage of malicious nodes. The graph shows clearly that as the number of malicious nodes increases in the network, the percentage of successfully established route decreases. This is mainly due to the fact that the higher the percentage of malicious nodes, the higher probability that these nodes will drop the route request messages, leading to a lower connectivity rate. The graph shows also the difference between route requests with HTR and LTR for the intermediate nodes: the difference in the degree of connectivity between HTR and LTR is mainly due to the higher trust requirement of HTR, which restricts the number of intermediate nodes that can participate in the routing protocol leading to a lower connectivity; LTR route requests uses intermediate nodes even if they have low trust value.

Figure 7 illustrates the routing overhead as a function of the percentage of malicious nodes for both HTR and LTR. The graph shows that as the number of malicious nodes increases, the number of times the source node has to run the route discovery algorithm increases, and hence an increase in the routing overhead

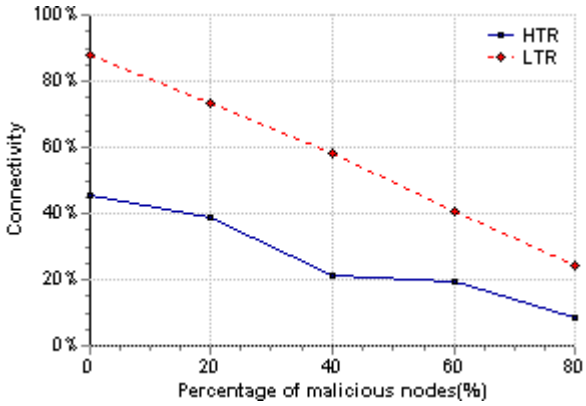

Figure 6. Connectivity vs. percentage of malicious nodes

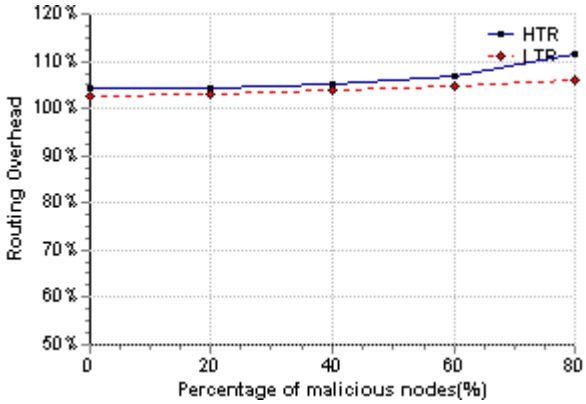

Figure 7. Routing overhead vs. percentage of malicious nodes

This is also the result of intermediate malicious nodes dropping the route request or reply messages, resulting in an increase in the number of route requests. HTR has also higher routing overhead, since the probability of route requests is higher because of the restriction on the trust level of intermediate nodes, hence requiring a larger number of route requests.

The results in Figure 8 show the change in route length as a result of the change in the percentage of malicious nodes in the network, and also for HTR and LTR. For both HTR and LTR, the average route length increases with the increase of the percentage of malicious nodes. This is mainly due to the fact that the shortest path might not be satisfactory in terms of the trust level requirement. The graph shows also that when there are few malicious nodes in the network, the average path length for the HTR is higher than the average path length for the LTR. This could be explained by the fact that the HTR adds more constraints on the path discovery algorithm, but since paths do exist between

communicating nodes, the path would eventually be found, but it will be longer.

Figure 9 shows the end-to-end delay for the HTR and LTR, as a function of the percentage of malicious nodes. As explained in the previous paragraph, a higher percentage of malicious nodes means a longer route between the source and the destination, and hence additional delay in the packet delivery. Additionally, since HTR

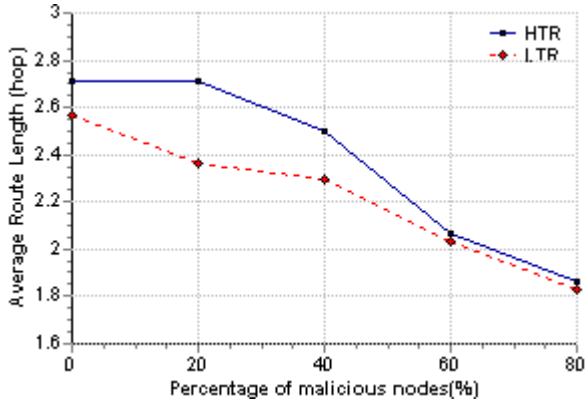

Figure 8. Average route length vs. percentage of malicious nodes.

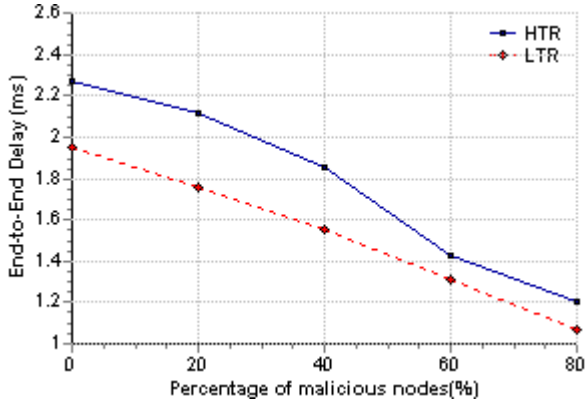

Figure 9. End-to-end delay vs. percentage of malicious nodes.

routes are longer than LTR routes, this means also higher end-to-end delay value.

Figure 10 and Figure 11 illustrate the difference between the number of packets prepared to be sent, the actual number of packets sent, and the number of packets that were received at the destination, as the percentage of malicious nodes in the network changes. As noticed from the graph, using the LTR generates a higher throughput, especially when the percentage of bad nodes is low; this is mainly due to the fact that all nodes are considered initially as malicious nodes, and their status is updated during the course of the simulation.

Figure 12 shows the number of exchanged UPDATE messages used to establish and update the trust relationship between neighboring mobile nodes. This number drops as the percentage of malicious nodes increases in the network. This number is expected to decrease, since nodes send only UPDATE messages when their status is changed from malicious (default) to good node. But as the percentage of malicious nodes increases in the network, the status of all the nodes remains malicious, fewer and fewer UPDATE messages are sent. The issue of coalition between malicious nodes is not included in our current work.

6. CONCLUSION

Security and anonymity are the most challenging issues in wireless and mobile ad hoc networks (MANET). In this paper, we

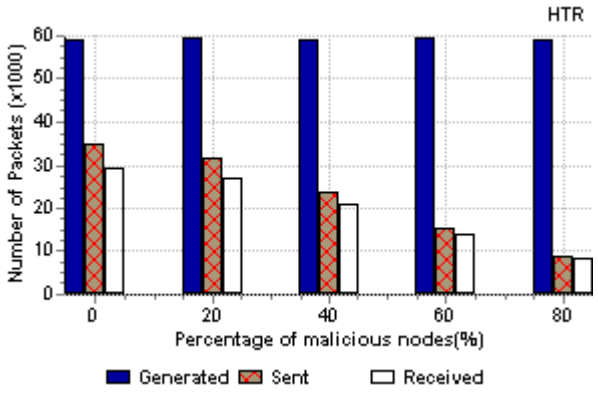

Figure 10. Number of packet for HTR vs. percentage of malicious nodes.

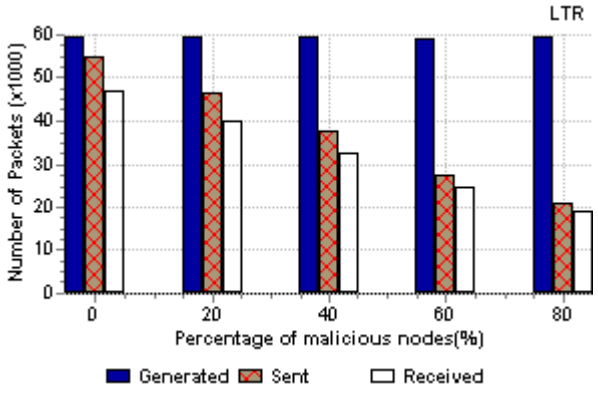

Figure 11. Number of packet for LTR vs. percentage of malicious nodes.

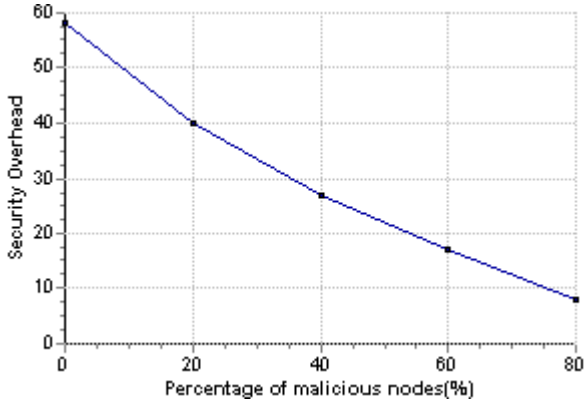

Figure 12. Security overhead vs. percentage of malicious nodes.

present an efficient secure distributed anonymous routing protocol for MANET, which anonymously creates routes dynamically to support onion routing without the originator knowing neither the keys of the mix nodes nor the topology of the network. We discuss the algorithm, and present an extensive set of simulation experiments to evaluate its performance analysis. Our simulation experiments indicate clearly that the anonymity in MANET is feasible, could be incorporated within an ad hoc routing protocol and provide a good solution for achieving anonymity in MANET at a reasonable additional cost when compared to the well-known DSR ad hoc routing protocol.

7. REFERENCES

[1] Anonymity on the Internet. http://www.sendfakemail.com/~raph/remailer-list.html, Accessed Jun. 2004.

[2] Boukerche, A., El-Khatib, K., and Xu, L. *Secure Routing Protocols for mobile Ad Hoc Networks*. Technical Report, TR-2004, University of Ottawa.

[3] Boukerche, A. and Notare, M. Neural Fraud Based Intrusion Detection for Mobile Phone Operations, *IEEE BioSP3*, 2001

[4] Boukerche, A. and Notare, M. Behavior Based Intrusion Detection in Mobile Phone Systems, *Journal of Parallel and Distributed Computing.* 2002.

[5] Broch, J., Maltz, D. A., Johnson, D. B., Hu, Y-C., and Jetcheva, J. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. ACM MOBICOM*, pages 85–97, Oct. 1998.

[6] Chaum, D. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM, 24.2,* (Feb 1981) 84-88.

[7] Chaum, D. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptography, 1.1,* (1988) 65-75.

[8] Electronic Frontiers Georgia (EFGA). Anonymous remailer information. http://anon.efga.org/Remailers/.

[9] Freedman, M. J., Morris, R. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the First International Workshop on Peer-to-Peer Systems* (Cambridge, MA, Mar. 2002).

[10] Goldberg, I., and Shostack, A. Freedom network 1.0 architecture, November 1999.

[11] Johnson, D. and Maltz, D. Dynamic source routing in ad hoc wireless networks. *T. Imielinski and H. Korth, editors, Mobile computing*, Kluwer Academic, 1996.

[12] Johnson, D. B., Maltz, D. A., and Broch, J. DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks. In *Ad Hoc Networking*, ch. 5, pp. 139-172. Addison-Wesley, 2001.

[13] Korba L., Song, R., and Yee, G. Anonymous Communications for Mobile Agents. *MATA 2002*: 171-181

[14] Lundberg, J. *Routing Security in Ad Hoc Networks*. Tech. Rep. Tik-110.501, Helsinki University of Technology, 2000.

[15] May, T. C. http://www2.pro-ns.net/~crypto/chapter8.html, Accessed June 2004.

[16] NS-2, available at http://www.isi.edu/nsnam/ns/

[17] Papadimitratos, P. and Haas, Z. J. Secure Routing for Mobile Ad hoc Networks. *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002),* San Antonio, TX, January 27-31, 2002.

[18] Perkins, C. E. and E. Royer, M. Ad hoc on demand distance vector (AODV) routing. http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-00.txt, 1997. IETF Internet Draft.

[19] Reed, M., Syverson, P., and Goldschlag, D. Proxies for anonymous routing. In *12th Annual Computer Security Applications Conference*, (Dec. 1995), 95-104.

[20] Reiter, M. K. and Rubin, A. D. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security, 1.1, (*Nov. 1998), 66-92.

[21] Rennhard, M. *MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection*. Technical Report Nr. 147, TIK, ETH Zurich, Switzerland, August 2002

[22] Sanzgiri, K., Dahill, B., Levine, B. N., Shields, C., and Belding-Royer, E. M. A Secure Routing Protocol for Ad Hoc Networks, In *Proceedings of 2002 IEEE International Conference on Network Protocols (ICNP),* Nov. 2002.

[23] Syverson, P. F., Goldschlag, D. M., and Reed, M. G. Anonymous connections and onion routing. In *Proceedings of the IEEE Symposium on Security and Privacy* (Oakland, California, May1997), 44–54.

[24] Venkatraman, L., Agrawal, D.P. Strategies for enhancing routing security in protocols for mobile ad hoc networks, in *Journal of Parallel and Distributed Computing, 63.2* (February 2003), Special issue on Routing in mobile and wireless ad hoc networks, Pages: 214 – 227, Year of Publication: 2003, ISSN:0743-7315

[25] Yi, S., Naldurg, P., and Kravets, R. Security-Aware Ad Hoc Routing Protocol for Wireless Networks. *The 6th World Multi-Conference on Systemics, Cybernetics and Informatics* (SCI 2002), 2002.