Available online at www.sciencedirect.com
SCIENCE DIRECT
Computer Communications 28 (2005) 1193–1203
computer communications
www.elsevier.com/locate/comcom

An efficient secure distributed anonymous routing protocol for mobile and wireless ad hoc networks ☆

Azzedine Boukerche[a,b,*], Khalil El-Khatib[a], Li Xu[a], Larry Korba[b]

[a] *PARADISE Research Laboratory SITE, University of Ottawa, Ottawa, Ont., Canada*
[b] *National Council of Canada, Ottawa, Canada*

Received 21 July 2003; accepted 21 July 2004
Available online 30 September 2004

Abstract

An ad hoc wireless network is a temporary and dynamic environment where a group of mobile nodes with radio frequency transceivers communicate with each other without the intervention of any centralized administration or established infrastructure. Due to the limited transmission range of each mobile node, communication sessions between two nodes are usually established through a number of intermediate nodes, which are supposed to be willing to cooperate while forwarding the messages they receive to their destination. Unfortunately, some of these intermediate nodes might not be trustworthy and might be malicious, thereby forming a threat to the security and/or confidentiality of the exchanged data between the mobile nodes. While data encryption can protect the content exchanged between nodes, analysis of communication patterns may reveal valuable information about end users and their relationships. Using anonymous paths for communication provides security and privacy against traffic analysis. To establish these anonymous paths, in a traditional wired network, nodes build a global view of the network by exchanging routing information, whereas in an ad hoc wireless network, building this global view is not an option. In this paper, we propose a novel distributed routing protocol which guarantees security, anonymity and high reliability of the established route in a hostile environment, such as ad hoc wireless network, by encrypting routing packet header and abstaining from using unreliable intermediate node. 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. We describe our protocol, and provide its proof of correctness.

Keywords: Wireless networks; Mobile Ad hoc networks; Routing algorithms; Wireless security

1. Introduction

Continuous advances in wireless and mobile communication technology coupled with the recent proliferation of portable computer devices have led development efforts for future wireless networks towards wireless and mobile ad hoc networks. In an ad hoc wireless network, two nodes can communicate directly as long as they are within the radio communication range of each other. However, since there is no stationary infrastructure such as base stations, each node has to act as a router for itself. A routing protocol for ad hoc networks is executed on every host and is therefore subject to the limit of the resources at each mobile host.

Ad hoc wireless networks have received a great deal of attention in recent years. This is mainly due to their potential applications ranging from military to non-military type of applications. 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.

☆ This work was partially supported by NRC, NSERC and Canada Research Chair Program Funds.
* Corresponding author.
E-mail address: bouketrch@site.uottawa.ca (A. Boukerche).

0140-3664/$ - see front matter © 2004 Published by Elsevier B.V.
doi:10.1016/j.comcom.2004.07.019

For the Internet, several network-based anonymity approaches provide anonymous communication between end-nodes. These approaches include DC-nets [1], Crowds [2], MIX networks [3], and Onion Routing [4]. 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 propose a novel secure distributed path construction protocol for anonymous communication and wireless ad hoc networks. As opposed to previous related protocols, the proposed 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, and only the selected neighbor nodes are able to decrypt it. 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 multi-cast mechanism and the layered encryption used in the protocol ensure the anonymity of the sender and receiver nodes.

The remainder of the paper is organized as follows: Section 2 will review previous and related work. Section 3 discusses the security issues in wireless ad hoc networks. Section 4 describes the trust management system upon which our algorithm relies upon. Section 5 describes our secure distributed anonymous routing protocol, which we refer to as SDAR. Section 6 presents the main characteristic of our protocol. Section 7 presents an analysis of our scheme. The conclusion follows in Section 8.

2. Previous and related work

There have been constant efforts made by researchers to increase the security of ad hoc routing protocols. In this section, since our scheme is based upon the anonymous communication paradigms, we will first review related routing algorithms for anonymous communication systems, and then review previous secure ad hoc routing schemes.

2.1. Anonymous communication in the onion routing protocol

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 [4] is a communication protocol that is resistance against some of these attacks. It employs a network of Chaum MIXes [3] 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 are 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.

During the connection setup phase, an initiating application makes a socket connection to an application specific proxy on one onion router. The proxy determines the route through the onion routing network using its knowledge of the network topology and available onion routers. To protect the data and routing information, the proxy constructs a multi-layer encrypted data structure called an onion and sends it through the network. Each layer of the onion defines the next hop in the route. An onion router that receives an onion peels off the topmost layer, identifies the next hop, and sends the remaining onion to the next router. In addition to carrying next hop information, each onion layer contains key seed material from which keys are generated for decrypting and encrypting data sent forward or backward along the anonymous connection.

Once the anonymous connection is established, the data transfer phase begins and data can be sent in both directions. The initiator's onion proxy receives data from an application, breaks it into fixed sized cells, and encrypts each cell multiple times—once for each onion router the message traverses on its way to the destination. As a cell of data moves through the network, each onion router removes one layer of encryption and forwards the cell to the next node along the path. Eventually, the data emerges at the final onion router in the path. At this point, the recipient's proxy regroups the cells into the data stream originally submitted by the application and, acting as the receiver proxy, forwards it to the destination application. For data moving backward, from the recipient to the initiator, the process occurs in the reverse order, with the recipient's proxy

breaking the traffic into cells, and successive onion routers encrypting the cells it for the return journey.

In the *connection termination phase*, the anonymous connection established in the connection setup *phase* is torn down. This involves the removal of encoded next hop information in each onion router making up the connection.

2.2. Finding anonymous paths in current anonymous communication systems

Over the Internet, anonymous systems [5–7] 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 [8] 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 [9,10] 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 [11] and MorphMix [12] have discussed the difficulties of constructing routes in dynamic environments. In Tarzan [24], the initiating node establishes the anonymous path by iteratively adding one node at a time to the path. In a single iteration, the initiator adds one node to the path, and receives the list of neighbors of that node. The initiator selects one of these neighboring nodes to be added to the path during the next iteration. A similar approach was used in MorphMiX [12], but the difference is that in MorphMix, and instead of the initiator, a trusted third party makes the selection of the next node. Using the probability of appearance of nodes on the path, the path initiator can, up to a certain degree, determine existence of malicious collusions among the nodes on the path. The problem with Tarzan and MorphMix is that it takes a long time to construct the paths, which is a major problem for dynamic environment, and wireless ad hoc networks.

2.3. Securing ad hoc networks routing protocol

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 [13,23,24]. A number of protocols have been developed to add security to routing in ad hoc networks. Papadimitratos and Haas [14] proposed Secure Routing Protocol (SRP) based on DSR [15,16]. The protocol assumes the existence of a security association between the source and destination to validate the integrity of a discovered route. Dahill [17] proposed the Authenticated Routing for Ad hoc Networks (ARAN) protocol that uses public key cryptography instead of the shared security association used in the SRP [14]. 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 [18] developed a generalized Security-Aware Ad hoc Routing (SAR) 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 [19] proposed an approach for enhancing the security of AODV protocol [20], which is based on public key cryptography. In their approach, two systems, External Attack Prevention System (EAPS) and Internal Attack Detection and Correction System (IADCS) 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).

3. Security issues in ad hoc routing

Security is a complex issue in ad hoc wireless networks. This complexity is the result of a number of factors, including the wireless medium, nodes mobility and lack of infrastructure. A malicious node (or attacker) can easily eavesdrop into the wireless communication channels and infer communication. Additionally, because of the mobility of the nodes and the absence of infrastructure, communicating nodes rely on other mobile intermediate nodes to

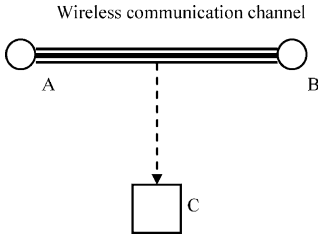

Fig. 1. A passive attack.

relay their data. This openness in ad hoc wireless networks makes the nodes more susceptible to attacks by hackers.

There are different kinds of attacks that can be used by malicious nodes (or users) to harm the network, and leave its ad hoc routing protocols unreliable. They can be basically categorized, based upon the nature of the attacks, into *passive attacks* and *active attacks* [21].

3.1. Passive attacks

A passive attack happens when an attacker un-intrusively taps on a communication channel between two nodes without disturbing the communication. Fig. 1 shows a schematic description of a passive attacker *C*, eavesdropping on the communication channel between *A* and *B*.

A passive attacker does not interrupt the operation of the communication channel. Instead, the primary purpose of the attack is to discover some valuable information about the data or control messages sent over the communication channel. While passive attack might not look harmful, it still can pose a threat to the security and privacy of the system. Traffic analysis can easily expose nodes' address or disclose the relationship between nodes; it can also reveal the topology of the network. For instance, if an eavesdropper finds out that a large volume of traffic is sent through a particular node, he can draw a conclusion that the node plays an important role in the network and might later mount an active attack against that node, leading to a major disruptive of the network.

3.2. Active attacks

An active attack typically involves an attacker's direct intervention with the data and/or control information sent based upon the size of the communication channel. As shown in Fig. 2, an active attacker *C* can listen, modify, and inject messages into the communication channel between *A* and *B*.

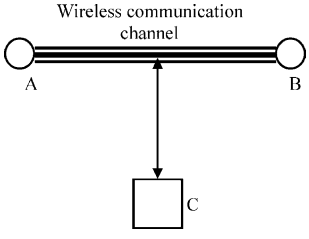

Fig. 2. An active attack.

Active attackers might maliciously participate in the routing protocol by replaying, modifying or deleting routing packets. For instance, in a *replay attack*, the attacker adversely re-sends a valid routing packet that was transmitted earlier, while in a *modification attack*, the routing information is maliciously modified and then transmitted. Active attacks on routing protocols usually lead to malicious updates on the routing table and hence packets being sent to false destination, or it might lead to the creation of routing loops and network congestion. Active attacks can be further classified into two groups, *external attacks* and *internal attacks* [19]. External attacks are caused by nodes that do not belong to the ad hoc network, while internal attacks are usually caused by compromised nodes belonging to ad hoc the network and hence are more difficult to protect against than external attacks.

4. Trust management system

As we mentioned earlier, due to the openness of ad hoc wireless environments, 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.

4.1. 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 [10] 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 broadcasting 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 the 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.

4.2. 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 $\delta 1$, while the second trust level, i.e. the medium level, contains the nodes whose trust level is between $\delta 1$ and $\delta 2$. The trust level, corresponding to the high level, contains the nodes whose trust value is between $\delta 2$ and Φ. Each node selects independently the values for $\delta 1$, $\delta 2$, and Φ.

The central 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 HTLCK) and Medium Trust Level Community Key (referred to as MTLCK), whereas, the neighbors in medium trust level have only MTLCK. 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. We will assume that the node assigns a medium trust level to a new neighbor and shares with it the MTLCK. 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.

4.3. 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. Appendices A and B give a detailed description of how a node can identify these two kinds of malicious behaviors of its neighboring nodes.

4.4. Trust-based distributed route selection mechanism

Our routing protocol, as we shall see in Section 5, 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.

5. A secure distributed anonymous routing protocol (SDAR)

In this section, we introduce 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.

5.1. 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.

5.2. 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.

In Table 1, we define the main notations used in this paper.

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

5.3. 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 it is 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 Fig. 3.

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

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

Fig. 3. *Path discovery* message just sent by the source M_s *is defined as H* (*TYPE, TRUST_REQ, TPK, TSK,* ID_R, K_S, ID_S, PK_S, $SN_{Session\_ID_S}$, PL_S, P_S).

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 ID_R 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 forth part and then use the retrieved session key K_S to decrypt the forth 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.

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

Fig. 4. Path discovery message just processed by node i M_S *is defined as* H(*TYPE, TRUST_REQ, TPK, TSK,* ID_R, K_S, ID_S, PK_S, $SN_{Session\_ID_s}$, PL_S, P_S), *while* M_{ID_i} *is as* 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* $node_{i-1}$.

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 Fig. 4.

5.4. 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 id's 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 will have the format shown in Fig. 5.

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

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

Fig. 5. Path reverse Message M_i *is defined as* $E_{K_{i-1}}(M_{i-2})$, $SN_{Session\_ID_i}$, $SN_{Session\_ID_{i-2}}H(M_{i-2})$, $H_{K_i}(N_i)$. N_i *is defined as* $(E_{K_i}(M_{i-1})$, $SN_{Session\_ID_i}$, $SN_{Session\_ID_{i-2}}$, $H_{K_i}(N_i))$. M_{S-1} *is a padding which has the same number of bits as any* M_j. $SN_{Session\_ID_{S-i}}$ *is a random number having the same number of bits as any regular* $SN_{Session\_ID_j}$ *and generated by the source node.*

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.

5.5. 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.

6. SDAR protocol characteristics

The proposed SDAR protocol has a number of characteristics, including Non-Source-Based Routing, Flexible and Reliable Route Selection and Resilience against Path Hijacking.

6.1. Non-source-based routing

The proposed protocol has a major advantage over the standard routing protocol for anonymous communication systems in the way routing paths are constructed in that it does not require a global view of the network topology. In standard onion routing [4], the source onion node must know in advance the topology and link state of the network before it can establish a routing path. The source onion node must also know the public keys of all onion nodes on the path as well as the exit policies for the edge onion nodes. In our protocol, each node in the network contributes toward the final routing path by forwarding the path discovery and path reverse messages. This approach eliminates the need for managing routing centrally.

Moreover, similar to other dynamic routing protocols, the proposed protocol gathers routing information only when the session is started or when the path breaks. Information about nodes that have joined or left the ad hoc network need not be propagated to all nodes. In standard onion routing this information is needed so that source onion nodes can build a routing path with viable nodes.

On the downside, the encryption/decryption at each node requires a fair amount of computational power to be viable. Current wireless nodes may not have sufficient computational capability for such processing.

6.2. No source control over route length

Unlike DSR [15,16], the source node in our protocol cannot set a limit on the maximum number of nodes on the path. A large number of nodes on the routing path can render the path too slow for real-time interactive applications. These applications usually have an upper limit on the network delay, which limits the number of intermediate nodes in the route. In addition, a large number of nodes in the path could also increase the failure rate of data delivery.

6.3. Resilience against path hijacking

During the path discovery phase, each intermediate node receives a path discovery message and forwards it to all nodes within its wireless transmission range While a well-behaved node forwards the message to all neighboring nodes in an unbiased way, a malicious node might forward the message only to its neighboring malicious nodes, resulting in a path with only malicious nodes. We refer to this situation as ‘path hijacking’.

The proposed protocol proves to be resilient against path hijacking. To confirm that, note that the protocol terminates successfully only after the trusted intended receiver triggers the path reverse phase, and after the path reverse message has made its way successfully to the source onion node. If malicious nodes keep on forwarding a path discovery message among themselves, the message will never get to the intended receiver and the source node will never get a path reverse message triggered by the path discovery message. Although in this case, the protocol may fail to return a suitable path, it is still resilient to path hijacking in the sense that the actual hijacking does not occur (note that other path discovery messages might still have made their way to the intended receiver and triggered a successful path reverse phase). If on the other hand, a malicious node decided to break the cycle and forward the message to a non-malicious node, and if this message gets to the intended receiver and initiates a path reverse message, the path will be constructed through a number of malicious nodes. But this case does not threaten the anonymity of the data traffic as was shown in [22], although it is a partial path hijacking. In any case, we only claim resilience to path hijacking, not immunity to it.

7. Security analysis

In this section, we will provide the proof of correctness of our SDAR protocol.

Theorem 1. *SDAR is secured against passive and active attacks, but not against Denial-of-Service attacks*

Proof.

1. SDAR provides protection against passive attacks. In the *path discovery* message, the identifier ID_S and public key

PK_S of the sender are encrypted with a one-time session key K_S. This key and the identifier ID_R of the intended receiver are encrypted with the public key PK_R of the intended receiver. The one-time public key encrypts the identities of intermediate nodes and the shared session keys. Thus, an adversary cannot find the real sender, receiver, and all intermediate nodes just by looking at the path discovery message. The same conclusions can be made for the *path reverse* message.

2. SDAR provides protection against active attacks like replay attacks and modification attacks. Using the session key K_S in the path discovery message, and the one-time public key *TPK*, the sender can discover a replayed path discovery message. Additionally, if some adversaries want to change the path discovery message or impersonate the sender or some intermediate nodes, the receiver can easily find out by verifying the signature since the sender and intermediate nodes have hashed the message, which is cumulative from its ancestor node, and signed the hashing value in the path discovery message.
3. A Denial-of-Service (DoS) attack would be a very dangerous attack on the protocol. The protocol itself does not provide a mechanism against this kind of attack. For instance a powerful adversary may simply flood the network with path discovery messages. Additionally, the small computational power on all wireless carry-on devices makes the protocol more vulnerable to this attack. This problem is common though to all routing protocols in ad hoc networks. □

Theorem 2. *SDAR maintains the anonymity of the sender and receiver.*

Proof.

1. During path discovery phase processing:
   - If all neighboring nodes of the sender joined in collusion, they would know which message originally came from the sender and which message was just forwarded by the sender, i.e. they would find the sender but will not know who the intended receiver is.
   - If all neighboring nodes of the receiver were in collusion together, they would know which message terminated in the receiver and which message was just forwarded by the receiver, i.e. they would find the potential receiver but would not know the identity of the sender.
   - If some intermediate nodes were in collusion together, they would only know that the message was forwarded. They, therefore, cannot confirm which node is the sender and which node is the receiver.
2. During path reverse phase processing:
   - If all neighboring nodes of the receiver joined in collusion, they would be able to determine who the receiver is. The collusion of all neighboring nodes can reveal the fact that the circled node is the node that started the path reverse phase, and hence it must be the intended receiver in the path discovery phase.
   - If all neighboring nodes of the sender were in collusion, they would be able to determine who the sender is. The situation is same as the above.
   - If some or all of the intermediate nodes were in collusion together, although they would know part of the path chain, they still would not be sure who the sender and receiver are since they would not know if the end node of the path reverse message is the sender or just another intermediate node, and the start node of the path reverse message is the receiver or just another intermediate node. □

Theorem 3. *SDAR is able to identify malicious nodes and avoid using them to establish routes.*

Proof. Due to the nature of ad hoc wireless network and the bi-direction of the link between two wireless nodes, a node can be always monitored by its neighboring nodes. Whenever a node behaves maliciously, its neighboring nodes will update their trust in the node. When a neighboring node receives later a request for a new path, it will avoid using the misbehaving node, and insure that the path is established with only trusted nodes. □

Theorem 4. *SDAR is able to establish a route matching certain trust requirement if enough nodes with qualifying trust value exist between the source and destination.*

Proof. During the path discovery phase, each intermediate node broadcasts the route request message to all of its neighboring nodes that satisfy the trust requirement set by the source node. Thus, starting from the source node, all the nodes with the requested trust level are passed the request message. These nodes in turn, pass the request message to neighboring trusted nodes. Therefore, any path of trusted nodes, starting at the source and ending at the destination node will be searched definitely be found. □

Additionally, even if the source node sets the trusted level for the route, each intermediate node is responsible for selecting the next hop on the path. Having each intermediate node selects its next hop insures that any malicious behavior by a node is observed by all neighboring nodes and will be reflected later during the route selection.

8. Conclusion

Security and Privacy are one of the most challenging issues in wireless and mobile ad hoc networks (MANET).

In this paper, we have presented a novel secure distributed anonymous routing protocol for MANET, which we refer to as SDAR. Our protocol 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 have shown that the protocol provides an adequate level of security and anonymity for both the sender and receiver during path establishment. Our approach has several advantages when compared to previous schemes that can be summarized as follow: (1) non-source-based routing—source node does not need to know global topology and link availability; route computation shared among many nodes; easy adaptability to changes in network topology (2) flexible and reliable route selection—route selection is based on the source node's trust requirement to the route and done in a distributed way in the path discovery phase according to intermediate nodes' own direct experience with its neighbor; and (3) resilience against path hijacking—resilience against malicious nodes compromising the communication through collusion.

In the future, we plan to implement our scheme for both proactive and reactive ad hoc routing protocols, and evaluate its performance using an extensive simulation set of experiments.

Appendix A. Finding malicious dropping behavior

During the path discovery phase, the unique identifier, *TPK*, is carried in every path discovery message and used to find malicious dropping simply by expecting to overhear the message with the same TPK from the neighboring node.

During the path reverse phase and data transfer phase, the same check is used, except that the $SN_{Session\_ID}$ is used instead of the *TPK*. In these two phases, a node's next hop's session key *ID* is obtained from path reverse message or data transfer message and locally stored by the node's predecessor. Looking at the message format in Fig. 5, we can see that $node_i$ can not only get its own session key ID, $SN_{Session\_ID_i}$, but also $SN_{\text{Session}\_ID_{i\text{-}2}}$, the session key *ID* of its next-next-hop node, $node_{i\text{-}2}$. The predecessor then expects that it should overhear the message carrying the session key *ID* of the node's next hop from the node.

Appendix B. Finding malicious modification behavior

We will show here how to identify malicious modification using node $Node_{i\text{-}1}$ in Fig. 5 as an example. When $Node_{i\text{-}1}$ receives the message, $M_{i\text{-}1}$, from $Node_i$, it processes $M_{i\text{-}1}$ according to the following steps:

1. $Node_{i\text{-}1}$ retrieves $SN_{Session\_ID_{i\text{-}1}}$ and $SN_{Session\_ID_{i\text{-}3}}$ from $M_{i\text{-}1}$, then it stores $SN_{Session\_ID_{i\text{-}3}}$ and gets its corresponding session key, $K_{i\text{-}1}$, through $SN_{Session\_ID_{i\text{-}1}}$, then,
2. $Node_{i\text{-}1}$ separates $N_{i\text{-}1}$, from $M_{i\text{-}1}$, and mixes it with $K_{i\text{-}1}$, and then hashes the mixture, afterwards,
3. $Node_{i\text{-}1}$ checks whether the hash result is equal to $H_{K_{i\text{-}1}}(N_{i\text{-}1})$ computed from $M_{i\text{-}1}$ or not. If it is not, $Node_i$ must have done *malicious modification* on $M_{i\text{-}1}$, and the checking stops; otherwise, the message, $M_{i\text{-}1}$, sent by $Node_i$ is not modified,
4. $Node_{i\text{-}1}$ gets rid of $SN_{Session\_ID_{i\text{-}1}}$ and $H(M_{i\text{-}3})$ from $N_{i\text{-}1}$, and store $H(M_{i\text{-}3})$ locally, then $Node_{i\text{-}1}$ decrypts the rest of $N_{i\text{-}1}$ with $K_{i\text{-}1}$, and sends the decryption result, $M_{i\text{-}2}$, to $Node_{i\text{-}2}$ with no modification.
5. $Node_{i\text{-}2}$ is supposed to do the same thing as what $\text{Node}_{i\text{-}1}$ did and to send $M_{i\text{-}3}$ to $\text{Node}_{i\text{-}3}$. After having overheard $M_{i\text{-}3}$ from $Node_{i\text{-}2}$, $Node_{i\text{-}1}$ hashes it and checks whether the hash result is equal to previously stored $H(M_{i\text{-}3})$ or not. If equal, the message, $M_{i\text{-}3}$, sent by $\text{Node}_{i\text{-}2}$ is not modified, and otherwise, $\text{Node}_{i\text{-}2}$ must have had *malicious modification* behavior on $M_{i\text{-}3}$.

References

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

[2] M.K. Reiter, A.D. Rubin, Crowds: anonymity for web transactions, ACM Transactions on Information and System Security 1 (1) (1998) 66–92.

[3] D. Chaum, Untraceable electronic mail, return addresses, and digital pseudonyms, Communications of the ACM 24 (2) (1981) 84–88.

[4] M. Reed, P. Syverson, D. Goldschlag, Proxies for anonymous routing, in: Proceedings of the 12th Annual Computer Security Applications Conference, IEEE, New York, 1995, pp. 95–104.

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

[6] I. Goldberg, A. Shostack, Freedom Network 1.0 Architecture, November 1999.

[7] P.F. Syverson, D.M. Goldschlag, M.G. Reed, Anonymous connections and onion routing, in: Proceedings of the IEEE Symposium on Security and Privacy, Oakland, California, May 1997, pp. 44–54.

[8] L. Korba, R. Song, G. Yee, Anonymous communications for mobile agents, MATA 2002; 171–181.

[9] http://www.sendfakemail.com/~raph/remailer-list.html.

[10] http://www2.pro-ns.net/~crypto/chapter8.html.

[11] M.J. Freedman, R. Morris, Tarzan: a peer-to-peer anonymizing network layer, in: Proceedings of the First International Workshop on Peer-to-Peer Systems, Cambridge, MA, March 2002.

[12] M. Rennhard, MorphMix: peer-to-peer based anonymous internet usage with collusion detection. Technical Report No. 147, TIK, ETH Zurich, Switzerland, August 2002.

[13] J. Lundberg, Routing Security in Ad Hoc Networks. http://citeseer.nj.nec.com/400961.html.

[14] P. Papadimitratos, Z.J. Haas, Secure routing for mobile ad hoc networks, SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), San Antonio, TX, January. TX, 2002.

[15] D. Johnson, D. Maltz, Dynamic source routing in ad hoc wireless networks in: T. Imielinski, H. Korth (Eds.), Mobile Computing, Kluwer Academic Publishers, Dordrecht, MA, 1996.

[16] D.B. Johnson, D.A. Maltz, J. Broch, DSR: the dynamic source routing protocol for multi-hop wireless ad hoc networks, in: Ad Hoc Networking, Addison-Wesley, Reading, MA, 2001, chapter 5, pp. 139–172.

[17] K. Sanzgiri, B. Dahill, B.N. Levine, C. Shields, E.M.A Belding-Royer, A secure routing protocol for ad hoc networks, in: Proceedings of 2002 IEEE International Conference on Network Protocols (ICNP), November 2002.

[18] S. Yi, P. Naldurg, R. Kravets, Security-aware ad hoc routing protocol for wireless networks, The Sixth World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2002), 2002.

[19] L. Venkatraman, D.P. Agrawal, Strategies for enhancing routing security in protocols for mobile ad hoc networks, Journal of Parallel and Distributed Computing 63 (2) (2003) 214–227 Special Issue on Routing in Mobile and Wireless ad hoc Networks, Year of Publication, 2003, ISBN 0743-7315.

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

[21] L. Venkatraman, D.P. Agrawal, A novel authentication in ad hoc networks, in: Proceedings of the Second IEEE Wireless Communications and Networking Conference, Chicago, September 2000.

[22] J. Raymond, Traffic analysis: protocols attacks, design issues, and open problems, in: H. Federrath (Ed.), Anonymity 2000, Volume 2009 of Lecture Notes in Computer Science, Springer, Berlin, 2000, pp. 10–29.

[23] A. Boukerche, Performance evaluation of on-demand routing protocols, ACM/Kluwer Mobile Networks and Applications, 2004.

[24] A. Boukerche, S. Rogers, GPS query optimization in mobile and wireless ad hoc networks, Proceedings of the Sixth IEEE Symposium on Computers and Communications, 2003 pp. 198–203.