SDAR: A Secure Distributed Anonymous Routing Protocol for Wireless and Mobile Ad Hoc Networks

Azzedine Boukerche[†], Khalil El-Khatib[‡,†], Li Xu[†], Larry Korba[‡]
[†] SITE, University of Ottawa
E-mail: {bouketrch,lxi}@site.uottawa.ca
[‡] National Research Council of Canada
{Khalil.el-khatib,Larry.Korba}@nrc.gc.ca

Abstract

Providing security and privacy in mobile ad hoc networks has been a major issue over the last few years. Most of the research work has so far focused on providing security for routing and data content, but nothing has been done in regard to providing privacy and anonymity over these networks. 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.

1. Introduction

Security and privacy are complex issues 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 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* [1].

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 multicast 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 on anonymous communication systems. Section 3 discusses the security issues in routing protocols for 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 features of our protocol, and the conclusion follows in Section 7.

2. Anonymous Communication Systems

Over the Internet, anonymous systems [2,3,4] 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

Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04)
0742-1303/04 $ 20.00 IEEE

the network topology, and make anonymous connections through a sequence of MIXes instead of making direct socket connections to other hosts. The authors in [5] 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 [6,7] 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.

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 [8] is a communication protocol that is resistance against some of these attacks. It employs a network of Chaum MIXes [9] 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.

Recently, Tarzan [10] and MorphMix[11] have discussed the difficulties of constructing routes in dynamic environments. In Tarzan [10], 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 [11] 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.

3. Securing Ad hoc Network Routing Protocols

Due to the nature of the wireless environment and the lack of predefined infrastructure [12,13], achieving secure routing in wireless ad hoc networks is a complex task A number of protocols have been developed to add security to routing in ad hoc networks. Papadimitriou and Haas [14] proposed SRP (Secure Routing Protocol) 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. Sanzgiri *et al.* [17] 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 [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 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 [19] proposed an approach for enhancing the security of AODV protocol [20], which is 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).

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 [20] 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 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 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.

4.4 Trust-Based Distributed Route Selection Mechanism

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.

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.

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. The main notation used in this paper are presented in Table 1.

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*.

5.1 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's 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 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

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 4.

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 $node_{i-1}$.

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

Figure 2. Path discovery message just processed by $node_i$.

5.2 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* should have 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.

5.3 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. Features of the SDAR Protocol

The proposed SDAR protocol has a number of features, including:

- **Non-Source-Based Routing:** In standard onion routing [8], 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.
- **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.
- **Resilience against Path Hijacking:** While a well-behaved node forwards the routing messages 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 each other, 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.

In addition to these features, the SDAP protocol has a number of properties. Due to space limitation, we will list these properties as theorems; the proof of these theorems can be found in [21].

Theorem 6.1: *SDAR is secured against passive and active attacks, but not against Denial-of-Service attacks*

Theorem 6.2: *SDAR maintains the anonymity of the sender and receiver.*

Theorem 6.3: *SDAR is able to identify malicious nodes and avoid using them to establish routes.*

Theorem 6.4: *SDAR is able to establish a route matching certain trust requirement if enough nodes with qualifying trust value exist between the source and destination.*

7. Conclusion

Due to the nature of the wireless environment and the lack of predefined infrastructure, achieving secure routing in wireless ad hoc networks is a complex task. A number of protocols have been developed to add security to routing in ad hoc networks. In this paper, we have presented a novel secure distributed anonymous routing protocol for MANET, which we refer to as SDAR. We have discussed the protocol and highlighted its main features, which include *(i) Non-source-based routing (ii) Flexible and reliable route selection*, and *(3) Resilience against path hijacking*.

Reference

[1] 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.

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

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

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

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

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

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

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

[9] D. Chaum: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM, vol. 24, no. 2, pages 84-88, Feb. 1981.

[10] 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, Mar. 2002).

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

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

[13] A. Boukerche, "Performance Evaluation of On-Demand Routing Protocols", ACM/Kluwer Mobile Networks and Applications, 2004.

[14] P. Papadimitratos and 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 27-31, 2002.

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

[16] D. B. Johnson, D. A. Maltz, and J. Broch, 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.

[17] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. 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, and R.Kravets: Security-Aware Ad Hoc Routing Protocol for Wireless Networks. The 6th 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, in Journal of Parallel and Distributed Computing, Volume 63, Issue 2 (February 2003), Special issue on Routing in mobile and wireless ad hoc networks, Pages: 214 – 227, Year of Publication: 2003, ISSN:0743-7315

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

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