Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2014, Article ID 212350, 10 pages
http://dx.doi.org/10.1155/2014/212350

Research Article

An Anonymous Routing Protocol with Authenticated Key Establishment in Wireless Ad Hoc Networks

Wei Yuan

State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academic of Science, 89-A, Minzhuang Road, Haidian District, Beijing 100093, China

Correspondence should be addressed to Wei Yuan; yuanwei1@126.com

Received 15 October 2013; Accepted 24 December 2013; Published 16 January 2014

Academic Editor: Hwa-Young Jeong

Anonymity is an important concern in wireless communications. Most anonymous routing protocols in ad hoc networks usually assume that a shared secret exists between the sender and the receiver. These protocols hide either the sender and the receiver or the intermediate nodes from the ad hoc network. Different from previous anonymous secure routing protocols, this paper proposes an anonymous routing protocol, ARAKE, which not only makes the sender and the receiver anonymous but also hides the intermediate nodes from the network simultaneously. To make the protocol more practical in dynamic network, ARAKE uses the public key to substitute the shared secret. In ARAKE, the receiver can authenticate the sender and gets a shared secret without extra key establishment processes. ARAKE can prevent packet analysis attack as well as most active attacks that are based on route information. The denial-of-service attack to specific session also can be restrained. The simulative results have shown that ARAKE outperforms a representative protocol ARAN in terms of both communication and energy overheads.

1. Introduction

Ad hoc networks play an important role in urgent communications. In this environment, when an intermediate node receives a communication packet, it may modify, fabricate, or drop it. Based on the information of the received packet, the malicious node may impersonate the original sender as well. In addition, an adverse node also may capture a node physically, gets the secrets from that node, and continues to attack others using its secrets. For example, in Figure 1, A is the sender, X is the receiver, N_1 and N_2 are adverse nodes, and the other nodes are normal intermediate nodes. N_1 could initiate both passive and active attacks once it receives a packet.

The early routing protocols, such as AODV [1] and DSR [2], mainly focus on how to route the packets in ad hoc networks efficiently, but the potential adversaries that may sneak into normal nodes are usually not taken into account. As a result, some malicious nodes may affect the whole network seriously by creating wrong packets or collecting the traffic information [3]. Thus, the issue of secure and anonymous routing [4–7] has received significant attention in recent years.

Zapata proposes a secure version of AODV, called SAODV [8], to prevent the black hole attack on AODV. In this attack, a malicious node acts as an intermediate node and advertises itself on the shortest path to the destination by sending a route reply message. In SAODV, when the intermediate node sends a route reply message to the source, the source node will send a quick further route request message to the neighbors of that intermediate node. The neighbor node will reply with further route reply message which contains the intermediate node listed in its route. If it does not, then that neighbor node is a malicious node. The approach adopted in SAODV is adequate for solving the black hole problem, but the communication overheads are increased greatly. Moreover, it fails to detect the wormhole attacks, in which the neighbor of that intermediate node is its partner. Naturally, we image a situation: if malicious nodes do not know the identity of the source and the destination nodes, and how these attacks are launched. Anonymous routing is to achieve this condition.

In 2005, Sanzgiri et al. proposed an authenticated routing protocol for ad hoc networks, called ARAN [9]. In this protocol, all the packets are signed with the private keys

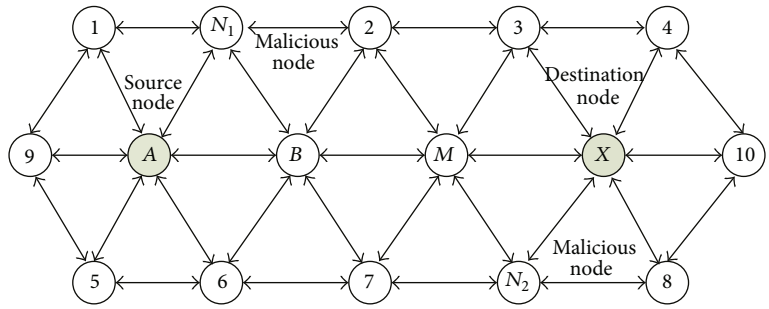

Figure 1: Example topology.

of the intermediate nodes. As a result, any modification of the packets can be found by neighbor nodes. ARAN divides the heterogeneous ad hoc network environments into three types: open, managed-open, and managed-hostile and describes the corresponding attributes needed in each environment. It solves the routing problem in open and managed-open environments, but the solution in managed-hostile environment is laid aside. Zhu et al. [10] propose an anonymous secure routing for ad hoc networks, which owns the anonymous properties in the field of identity, location, and route information. This protocol achieves anonymity of all participators. However, the basic assumption of its routing is that the source node and the destination node have to share secrets. In the ad hoc network, each node may be the source node or destination node in different sessions. To meet this assumption, each node has to share secrets with all the other nodes before the practical network is deployed. It is obvious that the storage of so many secrets is extravagant, which makes the network hard to manage. With the movement of the nodes, if some nodes leave the network, the corresponding secret stored in other nodes is useless. Furthermore, if some new nodes want to join this network, all the other nodes have to negotiate new secrets with them. That is nearly impossible for some deployed nodes. It means that this routing protocol is difficult to be scalable.

Most anonymous routing protocols [11–13] either hide the sender and the receiver [14] or hide the intermediate nodes [15]. Thus, these protocols can be applied in the environment that the nodes are safe physically. In battlefield environment, the exposed nodes may be annihilated easily. Inspired by the application demands described in ARAN [9] and ASR [10], in this paper, we contribute an anonymous secure routing protocol, ARAKE, for ad hoc networks. ARAKE hides the network topology information, such as the participators and their relationships. For example, in Figure 1, when N_1 receives a packet, it cannot get valuable information about which node is the sender, which node is the receiver, and which nodes relay the packet in terms of the packet contents. Further, it cannot distinguish whether the packet is request packet or reply packet. ARAKE is designed to prevent both passive and active attacks in the hostile environment. Different from most previous works, ARAKE does not rely on shared secret between the sender and receiver but constructs the routing by public key. Hence, the scalability of ARAKE is better than those protocols rely on shared secret. In addition, the receiver in ARAKE can authenticate the sender and gets a secret key from it without extra key establishment processes. Intermediate nodes in ARAKE do not need anonymous neighbor authentication before communications.

The remainder of this paper is structured as follows. Section 2 introduces the security assumptions and design goals. Section 3 describes the packet structure of ARAKE. Section 4 presents our anonymous routing protocol ARAKE. Section 5 provides a security analysis on ARAKE, while Section 6 evaluates ARAKE in terms of computation costs, communication costs, and energy costs. Finally, Section 7 concludes the paper.

2. Security Assumptions and Design Goals

2.1. Network Assumption. We consider that a node can move freely and thus may leave or join the network dynamically. The transmission range of a node is limited. Two nodes out of the direct transmission range communicate by means of some intermediate nodes. Each node has a fixed IP to represent its identity and a long-term public/private key pair to encrypt/sign the packet. The long-term public key of each node is authenticated by a central certificate authority (CA) and recorded in the public key certificate list of the CA.

Each node is capable of generating any number of temporary public/private key pairs for itself. Different from the long-term public key, the temporary public key is not bound to a node. Since the public key certificate is usually implemented with the RSA algorithm, in Section 6, ARAKE is implemented with the RSA algorithm [16] as well. Other public key encryption systems, such as ElGamal algorithm (based on discrete logarithm) and identity based cryptography (based on elliptic-curve discrete logarithm and pairings), can be used to replace RSA.

2.2. Adversary Assumption. Assume that adversaries can observe all the normal nodes simultaneously and eavesdrop on all the packets among them. Meanwhile, they can modify any packet they received, fabricate a new packet, and send it to normal nodes, but they cannot prevent a normal node broadcasting or unicasting the same packet that they received to other nodes. Adversaries can impersonate any nodes or rebroadcast the packets they received. In addition, adversaries can capture some but not all nodes in the network

so that the normal nodes are always the majority. Then they get the public/private keys of the captured nodes. The running environment and attackers' ability simulate the battlefield environment, and our assumptions are very near to the practical applications.

Based on its ability, adversaries can launch the following six attacks.

Packet Analysis Attacks. Due to the packet structure being open to attackers, they can analyze each packet and get useful information about packet number, receiving time, user identity, data content, or other traffic information. These attacks are passive and cannot be detected.

Redirection Attacks. The core of these attacks is to affect the route path, so that attackers can add themselves into the route path. To launch this attack, the active attacker may modify or delay the packet. Then he declares that he is in the shortest path between the source and destination nodes, and advertises himself to his neighbors. Redirection attacks may cause all the packets for some destination to be sent to the attacker or to a destination node outside the area (e.g., black-hole attack). In addition, attackers may initiate tunneling attack; that is, a pair of attacker nodes A and B linked via a private network connection. Each packet A receives from ad hoc networks is forwarded through the wormhole to B and is then retransmitted by B; similarly, B may forward all packets to A. As a result, attackers are recorded in the shortest path by other intermediate nodes.

Fabrication Attacks. Fabrications are the most common attacks. Adversaries may fabricate routing packets, the credit of other nodes, or the identity of some important node. Hence, the information that the receiver gets from that packet is wrong.

Reply Attacks. Adversaries record the valid packets and retransmit them to the other nodes. Then the receiver will generate more than one reply packet.

Impersonation Attacks. Adversaries initiate a session by impersonating the identity of an authorized node to gain access to network resources or to disturb the normal functioning of the network. With the help of man-in-the-middle attack, adversaries may even alter the information transmitted between two nodes and let them believe they are talking directly with each other.

Denial of Service Attacks. Adversaries try to disturb the communication in ad hoc network by flooding the network with vast amount of packets constantly. Although these packets are useless, each node that receives those packets must handle them. It causes services offered by the network not to work as usual, slow down, or even stop. Ad hoc networks are easier to be affected than wired networks, because there are more possibilities to perform such an attack.

Physical Attacks. In hostile environments, adversaries can easily annihilate any node if the location of that node is revealed to attackers. The physical attack is the strongest attack to nodes in ad hoc networks.

2.3. Design Goals. The goal of ARAKE is to resist the above attacks under our network and adversary assumption. First, since eavesdropping is easy to be achieved, all the fields of the route packet should be encrypted so that passive attack cannot get any useful information, even if all the packets of a session are collected by the adversary. Second, the routing protocol should be robust to node compromise. That is to say, if some nodes are compromised, the security of other nodes should not be harmed. Third, the protocol should be easily scalable. When the nodes increase, the protocol needs to adapt the mutative topology structure automatically. Fourth, the computation and communication costs should be cost-effective.

3. Packet Structure of ARAKE

The reply packet has the same format as the request packet, and their format is as follows:

$$[label, session_{prikey}, next_{pubkey}, sequence, dIP, label^{-}, sIP, source\_signature, source\_pubkey].$$

The first field is a unique session id. In the route request phase, the second field is the temporary public key of the node that relay the packet, and the third field is not used. In the route reply phase, the second field is the session private key, and the third field is the next node's temporary public key. The data in these two fields are encrypted, and their function is to find an anonymous link among intermediated nodes.

The fourth field records a random number shared between the source node and the destination node. The fifth field records the IP of the destination node. The sixth field records the session private key. The seventh field records the IP of the source node. The eighth field records the signature of the source node's IP address. The ninth field records the public key of the source node. All the data in fields 4–9 are encrypted by the receiver's long-term public key.

Each node sets up a route table and a spare route table after it receives a valid reply packet. These two tables have the same structure. Route table is to record the preferred neighbor node, which can relay the packet, and spare route table is to record all the other neighbor nodes.

The route table contains 4 fields:

$$[label^{+}, label^{-}, previous\_node, next\_node].$$

The first field records a session id and indicates which session that node is participating in. The second field records the paired session private key. The third field records its previous node's temporary public key. Similarly, the fourth field records its next node's temporary public key.

4. Proposed Protocol

In this section, we introduce the details of ARAKE. The variables and notations are listed in Table 1. ARAKE consists of four phases: system initialization, route request, route reply, and route maintenance. To describe the protocol clearly, we take the topology of Figure 1 as the example.

Table 1: Variables and notations.

K_U^+	Public key of node U
K_U^-	Private key of node U
TK_U^+	Temporary public key of node U
TK_U^-	Temporary private key of node U
$\{d\}K_U^+$	Encryption of data d using key K_U^+
$\{d\}K_U^-$	Decryption of data d using key K_U^-
$[d]K_U^-$	Signature of data d by node U
$[d]K_U^+$	Designature of data d by node U
N_U	Random number selected by node U
n_U	Random number selected by node U
IP_U	IP address of node U
label^+	Session public key
label^-	Session private key

4.1. System Initialization. In this phase, a central certificate authority S takes charge of selecting the key size and cipher size of the RSA algorithm and computes relevant public parameters, such as the big primes p, q, and $\varphi(n)$. Then the server generates the public key certificate for each node. Each node needs to download associated parameters. The format of the public key certificate is as follows:

$$Certificate_U : IP_U, K_U^+ \left[IP_U, K_U^+\right], K_S^-. \quad (1)$$

All nodes need to store the public key, K_S^+, of the CA, and each one can watch the CA's public key certificate list and download its public key certificates. CA defines the storage space of the route table and space route table that each node needs to obligate for ARAKE. The period to delete the incomplete items (e.g., the fourth field is null) in the two tables is defined by CA.

4.2. Route Request Phase

4.2.1. Source Node Request. Assume the source node A wants to set up a route to the destination node X. First, it selects the session private key, $label^-$, and generates the paired session public key $label^+$; for example, $label^+ \times label^- = 1 \bmod \varphi(n)$ in RSA. Then, A selects its temporary private key TK_A^- and computes the paired temporary public key TK_A^+ in the same way as $label^+$ and $label^-$. Finally node A broadcasts the following request packet to its neighbors:

$$A \longrightarrow broadcast : \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9\} \quad (2)$$

$A_1 \leftarrow label^+$, $A_2 \leftarrow \{TK_A^+\}label^+$, $A_3 \leftarrow n_a$, $A_4 \leftarrow \{N_A\}K_X^+$, $A_5 \leftarrow \{IP_X \oplus N_A\}K_X^+$, $A_6 \leftarrow \{label^- \oplus N_A\}K_X^+$, $A_7 \leftarrow \{IP_A \oplus N_A\}K_X^+$, $A_8 \leftarrow \{[IP_A]K_A^-\}K_X^+$, $A_9 \leftarrow \{[K_A^+]K_S^- \oplus N_A\}K_X^+$.

4.2.2. Intermediate Node Validation and Rebroadcast. Let B be an intermediate node which has received request packet from A. It needs to determine whether it rebroadcasts the packet. Only when the following conditions are satisfied, B rebroadcasts that packet.

(1) It is not the destination node. To verify this condition, B decrypts A_4 and A_6 with its private key K_B^-, selects a random number m, and validates whether (3) holds or not:

$$m^{A_1 \times \{\{A_6\}K_B^- \oplus \{A_4\}K_B^-\}} = m \bmod n. \quad (3)$$

If B's public key K_B^- was equal to K_X^-, then $\{A_6\}K_B^- = label^- \oplus N_A$, and $\{A_4\}K_B^- = N_A$. Equation (3) would hold. However, since $K_B^- \neq K_X^-$, the verification fails. Thus, B is not the destination node.

(2) That packet is not a reply packet. Since intermediate nodes also need to estimate the packet type in route reply phase, we introduce how to estimate the packet type later.

(3) It has not received other request packets with the same session id. To verify this condition, B only needs to query its route table.

If the above conditions are satisfied, B records A_1 and A_2 to the first and third fields of its route table, generates its temporary public/private key TK_B^+/TK_B^-, and broadcasts

$$B \longrightarrow broadcast : \{A_1, B_2, B_3, A_4, A_5, A_6, A_7, A_8, A_9\} \quad (4)$$

$B_2 = \{TK_B^+\}label^+$, and $B_3 = n_b$.

4.2.3. Destination Node Validation. After receiving the route request packet, the destination node X needs to verify that packet, and then reply to it. Assume that the packet to X is from an intermediate node M and has the format as $\{A_1, M_2, M_3, A_4, A_5, A_6, A_7, A_8, A_9\}$, $M_2 = \{TK_M^+\}label^+$, and $M_3 = n_m$.

X first confirms that it is the destination node in the same way as (3). Since the destination node is X, (3) turns into the following equation. It is easy to see that the following equation holds:

$$m^{label^+ \times \{\{\{label^- \oplus N_A\}K_X^+\}K_X^- \oplus \{\{N_A\}K_X^+\}K_X^-\}} = m \bmod n. \quad (5)$$

Then, besides $label^+$, X gets $XA_4 = N_A$, $XA_5 = IP_X$, $XA_6 = label^-$, $XA_7 = IP_A$, $XA_8 = [IP_A]K_A^-$, and $XA_9 = [[K_A^+]K_S^-]$ successively with its private key K_X^-. The source node's public key K_A^+ can be extracted by computing $K_A^+ = [XA_9]K_S^+$. Next, X compares XA_5 with its IP address and checks whether the following holds or not:

$$XA_7 = \left[XA_8\right]K_A^+. \quad (6)$$

If any one of the above verifications fails, X drops the packet and waits for the next one it can confirm. Otherwise, the packet is legal and then X generates a reply packet. At last, X decrypts M_2 to get TK_M^+ with $label^-$ and records it to the third field of its route table and spare route table. Figure 2 shows the flow chart of ARAKE.

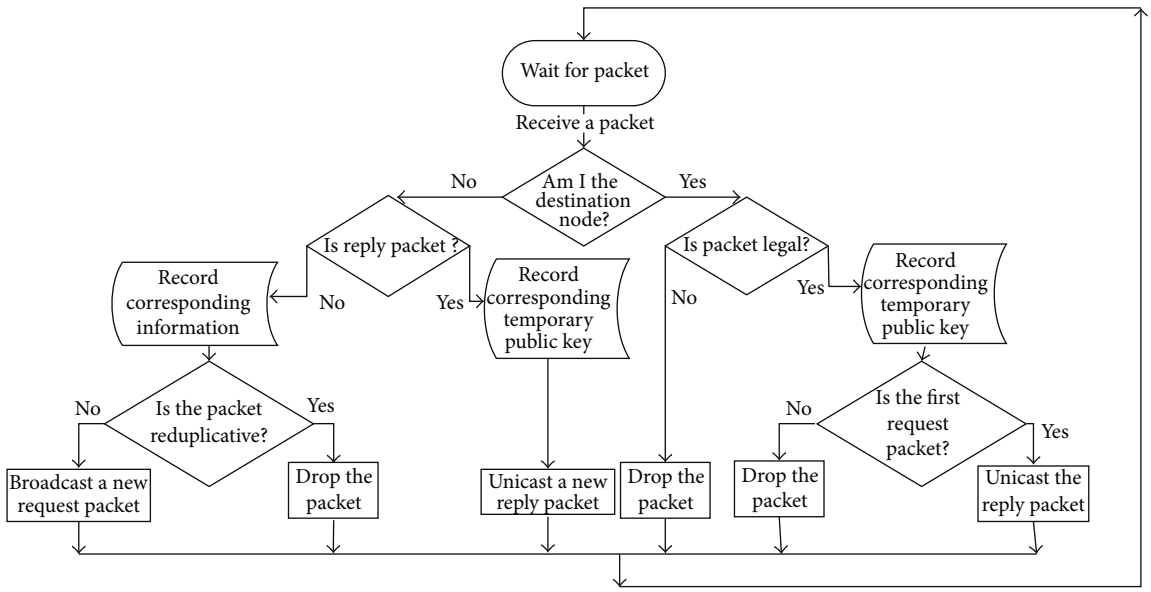

Figure 2: Flow chart of ARAKE.

4.3. Route Reply Phase

4.3.1. Destination Node Reply. Since X may receive more than one legal request packet, multiple temporary public keys with the same session id may be recorded in its spare route table. However, it only replies the first one and adds its temporary public key to the route table. Others are stored in the spare route table. To generate a reply packet, X selects its temporary private key TK_X^-, computes the temporary public key TK_X^+ in the same way as node A, and unicasts a reply packet back to M as follows:

$$X \longrightarrow M : \{X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9\} \quad (7)$$

$X_1 = label^+$, $X_2 = \{label^-\}TK_M^+$, $X_3 = \{TK_X^+\}TK_M^+$, $X_4 = \{N_A\}K_A^+$, $X_5 = \{IP_A \oplus N_A\}K_A^+$, $X_6 = \{label^- \oplus N_A\}K_A^+$, $X_7 = \{IP_X \oplus N_A\}K_A^+$, $X_8 = \{[IP_X]K_X^-\}K_A^+$, $X_9 = \{[K_X^+]K_S^- \oplus N_A\}K_A^+$.

4.3.2. Intermediate Node Validation and Reply. Upon receiving the reply packet, M needs to determine whether it should relay that packet. If the following two conditions hold, M generates a new reply packet to its previous node.

(1) It is not the destination node. The verification equation is the same as (3). Since the destination node is not M, the verification fails.

(2) That packet is a legal reply packet. To confirm this condition, M selects a random number m, computes $XM_2 = [X_2]TK_M^-$, and verifies whether the following holds or not:

$$m^{X_1 \times XM_2} = m \bmod n. \quad (8)$$

If the verification fails, M drops the packet. Otherwise, M confirms only that the packet is a legal reply packet. For this situation, the values of X_2 and X_3 are encrypted by M's temporary public key, TK_M^+, and X_2 are filled by the session private key $label^-$ simultaneously.

In the route request phase, the request packets are broadcasted in the network, and M may receive many request packets from different nodes. However, in the route reply phase, M only replies to the first one and adds its temporary public key to its own route table. Other nodes' temporary public keys are recorded to its spare route table. Assume that M receives the first route request packet $\{A_1, B_2, B_3, A_4, A_5, A_6, A_7, A_8, A_9\}$ from the node B and has recorded A_1 and B_2. Then M computes $BM_2 = \{B_2\}label^- = TK_B^+$, $XM_3 = \{X_3\}TK_M^- = TK_X^+$ and records BM_2 and XM_3 to the third and fourth fields of its route table. Finally, it unicasts a reply packet back to B:

$$M \longrightarrow B : \{X_1, MB_2, MB_3, X_4, X_5, X_6, X_7, X_8, X_9\} \quad (9)$$

$MB_2 = \{label^-\}TK_B^+$, and $MB_3 = \{TK_M^+\}TK_B^+$.

4.3.3. Source Node Validation. Besides the equations X should verify, A needs to verify whether the random number N_A decrypted from the packet is the original one it generated. Then N_A can be used to encrypted the data packets as the symmetric key. After the packet is confirmed, A adds TK_B^+ to the fourth field of its route table and spare route table.

4.4. Route Maintenance.

Each node deletes the incomplete items in its route table and spare route table periodically so that the two tables have enough space to record new items. A current route may be broken into two cases. First, a node in the route cannot receive packets from its neighbor nodes over a fixed time. Second, a node broadcasts a route block message actively. In the first case, the node selects another temporary public key in the next position of its spare route table to repair the route. In the second case, the node sends a ERR message to report that an active link will break off. Suppose that node M

cannot continue to keep the link between A and X; it unicasts the ERR message to its neighbors as follows:

$$M \longrightarrow B : label^{+}, \{ERR, label^{-}\} TK_{B}^{+},$$
$$M \longrightarrow X : label^{+}, \{ERR, label^{-}\} TK_{X}^{+}. \quad (10)$$

Upon receiving the packet, B decrypts it, checks the validity with $label^{+}$ and $label^{-}$, and then checks whether it can repair this route by querying its spare route table. If it cannot repair it, B continues to relay the corresponding message to A:

$$B \longrightarrow A : label^{+}, \{ERR, label^{-}\} TK_{A}^{+}. \quad (11)$$

If all the nodes in the route path cannot repair the route, either the source node A or the destination node X can initiate a new route request to set up a new link.

5. Security Analysis

In this section, we analyze the security of ARAKE against packet analysis attack, fabrication attack, redirection attack, impersonation attack, reply attack, and denial of service attack under the adversary assumption described in Section 2.

In ARAKE, all the request and reply packets have the same format. A node without special knowledge cannot distinguish the packet type directly. Only the source node and the destination node use their public keys in ARAKE. The authentication information is included in the 4–9 fields, and each field is encrypted by the public key of the destination node. Hence, these data are useless to all the other nodes. The data in the fifth, sixth, seventh, and ninth fields are confused by a random number N_A, and that number is encrypted and given in the fourth field. Owed to N_A, the ciphertexts generated by the same plaintext and public key are different in two sessions. As a result, attackers cannot guess which node generates the packet by comparing public keys. Since temporary public keys in the second and third fields are not associated with the node identity, attackers cannot deduce which node relays the packet from the temporary public keys.

Besides the format and content, a packet also provides another message, packet receiving time, to attackers. If attackers are not in the route path, they cannot distinguish the packet type, and then the receiving time is useless for them. If an attacker happens to be in the route path, he can record the receiving time of the reply packet and calculate the time delay from the time that he rebroadcasts the request packet to the time that he receives the reply packet. In addition, he could calculate the time delay between each neighbor node and him as well. Then he may estimate the general hops from the destination node to him. However, since the processing time of each node is not same, the estimated hops are not precise. To get more exact information, he needs to be in the route path of the same destination node for many times. However, since the identity and statuses of the destination node and the nodes out of his transmission range are anonymous for him, this condition is very hard to meet. As a result, the time analysis does not lead to serious threats. After the route reply phase finishes, the attacker can estimate general hops from the destination node to him and knows how many neighbor nodes in his transmission range. However, which nodes are on the route path and which nodes are closer to the destination node are unknown to him.

In the route request phase, all the intermediate nodes do not know the session private key. Thus, no information about the other nodes reveals in this phase. When the request packet arrives at the destination node, it should be verified by some equations. If the value of the fifth field is modified, the decrypted destination IP value cannot be the same as the receiver's IP. If any one of the seventh, eighth; or ninth field is modified, (6) will fail. The values in the first and sixth fields can mutually verify. If any one of the two fields is modified, the node will drop that packet directly. Only all of these verifications hold, the destination node believes that the packet is not modified by a malicious node. Otherwise, it just needs to wait for another valid packet. Similarly, the reply packet also needs these verifications. Hence, the fabrication attack cannot lead to serious problems.

To redirect a path from a normal node to the attacker, such as black hole attack and tunneling attack, the attackers or the cooperative attackers need to make the neighbors feel that they can relay the packet to the destination nodes with the shortest time. However, since the destination node and the intermediate nodes are all anonymous, the attackers cannot generate effective packets to affect neighbors' judgments. Thus, the redirection attack is invalid in ARAKE.

Similarly, the attacker cannot activate an impersonation attack, because he does not know who he should impersonate.

Each node in ARAKE only rebroadcasts the first request packet and unicasts the reply packet for one time. Repetitive packets are just dropped. Hence, the reply attack is invalid.

The most general threat to all routing protocols is the DoS attack. ARAKE cannot prevent this attack to the whole network, because the source node and the destination node are anonymous to all the other nodes. If a malicious node could generate a request packet to an inexistent destination node, the other nodes will receive this request packet and have to rebroadcast it. But ARAKE can protect a specific session from the DoS attack. In this situation, the source node cannot be the malicious node. No matter how many request packets it received, each node only generates one request packet in route request phase. In route reply phase, each node can only recognize the reply packet encrypted with its temporary public key. Hence, the valid packets won't be received except the neighbor nodes of the malicious node.

If the attacker happens to be in the route path and refuses to relay the packet to his neighbor, the link is broken. Then the node in the previous or next position may choose another node in their spare route table to replace the attacker as described in the route maintain process. It means that the attacker cannot launch DoS attack constantly.

6. Performance Evaluation

In this section, we evaluate the performance of ARAKE by comparing its computation costs, communication costs, and energy costs with ARAN.

6.1. Computation Performance. We first compare the computation overheads of ARAKE and ARAN on the nodes in the route path. Then we discuss the computation costs on the nodes out of the route path. Finally the influence to the network made by the malicious nodes is analyzed.

6.1.1. Computation Costs of Route Nodes. Due to the computation time of encryption being far greater than some fundamental operations (e.g., $\oplus$), we ignore these simple operations. In public key cryptogrammic system, the operation time of encryption and signature is nearly the same, and the operation time of decryption and designature is nearly the same, such as RSA [16] and BF-IBE [17]. Since we implement the ARAKE and ARAN with RSA, we make RSA as an example in this paper. The computation costs of encryption, decryption, signature, and designature in RSA are the same. We called all these operations basic operation. Measured by the basic operations, we compute the total computation times of the two protocols when the final route path consists of t nodes.

If there are t nodes, $1 + 3 + 5(t - 3) + 4 = 5t - 7$ basic operations are needed in route request phase and the same times of operation operations are needed in route reply phase of ARAN. So $10t - 14$ basic operations are needed in ARAN. In the same situation, $8 + 4(t - 2) + 8 = 4t + 8$ basic operations in route request phase and $9 + 5(t - 2) + 9 = 5t + 8$ basic operations in route reply phase are needed in ARAKE. So $9t + 16$ basic operations are needed in ARAKE. Thus, we know that the computation cost of ARAKE and ARAN is in the same level and the ARAKE is slightly better than ARAN when the number of route nodes is larger than 30 and without malicious nodes.

6.1.2. Computation Costs of Other Intermediate Nodes. In the route request phase of ARAKE, each node has to test whether it is the destination node. This needs 2 basic operations. For the reduplicative packets with the same label, the node can drop it directly. Hence, the computation times of the extra operations are decided by the number of the packets in this phases. In the route reply phase of ARAKE, each node also needs to confirm whether it is the source node or in the route path. Hence, the situation is similar to the route discovery process. In the route request phase of ARAN, each node should design the first part of the received packet to verify whether it is the destination node. Each node needs 4 basic operations except the neighbor of the source node. So $4n - 2m$ designature operations are needed, where m denotes the numbers of neighbors of the source node. The same computation times are needed in the route reply phase of ARAN. That is to say, the computation costs of these two protocols are still in the same level. A fact should be noticed that the source and the destination nodes need less computation costs than the intermediate nodes in ARAN. However, in ARAKE, the computation costs in the source node and the destination node are more than the intermediate nodes. The differences in the distribution of computation costs may lead to small distinction in communication performance of the two protocols.

6.1.3. Influence by Malicious Nodes. Suppose that one or more nodes except the route nodes are malicious. They send useless or fabricated packets to their neighbors. In ARAKE, each normal packet is encrypted and only designated node can decrypt it. So the malicious nodes can send their packet freely; it can only cause 1 time useless decryption to all of its neighbors. Since the neighbor nodes of the malicious node cannot recognize the fabricated packets, they cannot retransmit these packets. Hence, other nodes are not impacted by the malicious node. If a malicious node is just selected to relay the packet between the source node and the destination node, the only attack it can do is to drop the packet. It will not add the computation cost of other nodes. But other nodes should waste much time repairing the link. ARAN protocol cannot prevent the denial-of-service attack. When a node receives a packet, it will retransmit that packet to its neighbor node, and finally the packet will be received by all the nodes in the network. So the malicious node sends one packet to its neighbors; the number of the packets generated by the other nodes is equivalent to the total packets that those nodes set up a new route path.

6.2. Communications Costs. In this section, we implement ARAKE and another representative protocol, ARAN, with C++ language and compare their performance on the NS2. For both ARAN and ARAKE, the 1024 bits RSA algorithm is utilized to encrypt/decrypt or sign/design route packets. The transmission range of each node is 250 m. All the nodes are distributed at the vertex of a grid when the simulations start. The edge length of the grid is 200 m. The MAC layer protocol is IEEE 802.11. ARAKE and ARAN are simulated under 7 different configurations: 5 nodes lined up in a row; 10 nodes lined up in two rows with 5 nodes in each row; 15 nodes stand in three rows; 16 nodes stand in three rows; 25 nodes arrange in a square; 36 nodes arrange in a square; 49 nodes arrange in a square. The two protocols are evaluated by following four metrics.

(1) Average end-to-end delay of data packets: this is the average data packet delay from the source node to the destination node. For one packet, the delay time starts when data packet leaves the source node and ends at the time the destination node receives that data packet.

(2) Average route acquisition latency: this is the average route packet delay between the source node and destination. In ARAKE, the source node should add all possible paths into its spare routing table and then select the fast path to the routing table. Hence, the route is set up after all the reply packets back to the source node. The delay time starts from the first request packet sent by the source node and ends with the last receipt of corresponding route reply packet.

(3) Routing packet load: this metric shows the packet overhead to set up a route path between the source node and the destination node. We compute the total number of control packets and the average number of

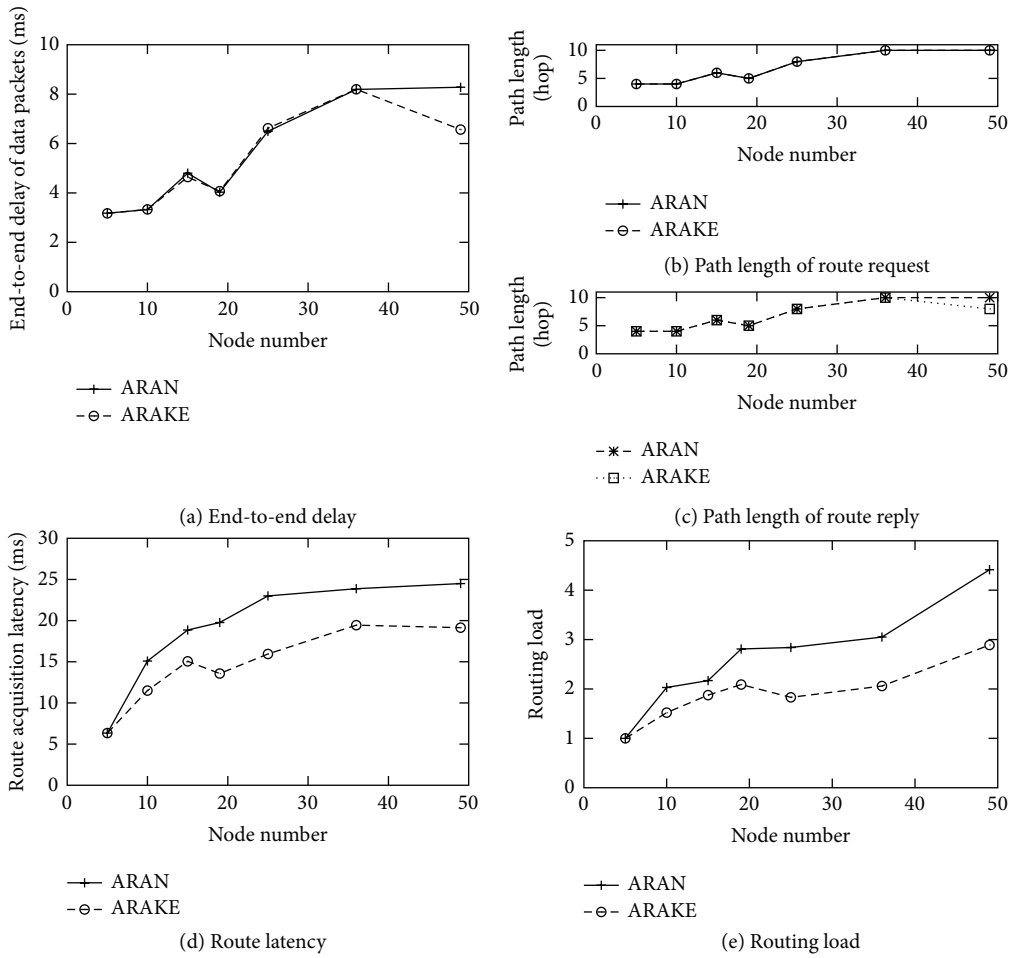

Figure 3: Communications costs.

data packets to send a data and get the value of this metrics by computing their quotient.

(4) Average path length: these metric is measured by the average hops between the source node and the destination node. We describe it with the hops taken by data packet from the source node to destination as well as from the destination node back to source.

Figure 3 shows the observed results of ARAN and ARAKE in different numbers of nodes. The end-to-end delay and the path length of the two protocols are nearly identical; the route acquisition latency and the routing load of ARAKE are slightly lower than ARAN.

The end-to-end delay and the path length of the two protocols are nearly identical except the scene that the node number is 49. In this scene, the two protocols happen to choose different paths and the path length is different as well. We describe it with Figure 4. The node 37 is the source node, and the node 13 is the destination node. In ARAN, the path from the source node to destination is different with the path from the destination node back to the source. It is occasional that the path length of route discovery is longer than route setup and the route setup path of the two protocols is identical at the same time. Hence, the data packets in ARAN need more time from the source node to destination. The average end-to-end delay is highly relevant with the path length. Hence, the values of the two protocols are different in this scene.

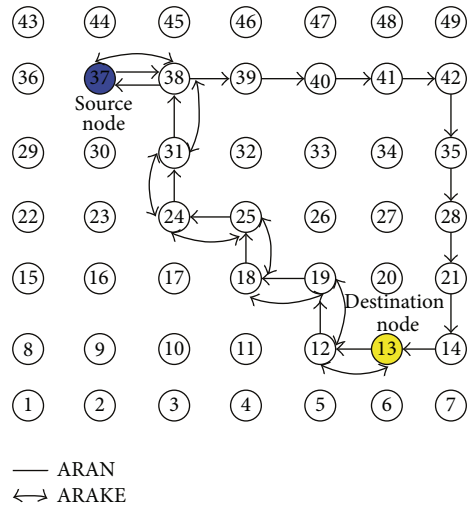

Figure 4: Route path in a 49-node scene.

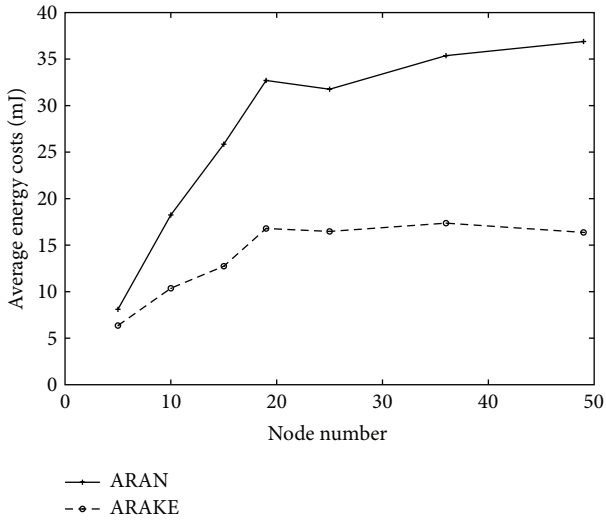

Figure 5: Energy costs.

In most scenes, the time that the route request packet arrives to the destination node and the first reply packet back to the source node are very near. However, the time that the last reply packet back to the source node are different between ARAKE and ARAN. Since the redundant paths have been saved in the spare route table, the source node in ARAKE only receives one reply packet. However, the source node receives more than one reply packets in ARAN. Hence, the route acquisition latency of ARAN is longer than ARAKE. This result is made by the broadcast nature of wireless transmissions. Due to more than one reply packets being sent back to the source node, the total route packets in ARAN are more than that in ARAKE. Hence, the routing load of ARAN is higher than ARAKE.

6.3. Energy Costs. The energy costs of ARAKE are highly related to the basic cryptographic operations. In ARAKE, each intermediate node needs to validate that it is not the destination node with (3), which needs 2 decryption operations. Then it needs 1 decryption operation to determine the packet type with (8). Hence, receiving a packet in ARAKE needs 3 decryption operations. In addition, generating a request packet needs 1 encryption operation and generating a reply packet needs 2 encryption operations. In ARAN, to receive a packet, a node needs to design two certificates to extract the public keys of the last two nodes with the server's public key and then designs the packet with these two public keys. Hence, 4 designature operations are needed to receive a packet. Finally, 1 signature operation is needed to generate a new packet in ARAN.

Suppose that each cryptographic operation needs to consume 1 mJ. Each intermediate node in ARAKE needs 3 mJ to receive and average 1.5 mJ to send a packet. Meanwhile, node in ARAN needs 4 mJ to receive and 1 mJ to send a packet. The idle energy cost is set to 0 mJ. Figure 5 shows the comparisons of their energy costs. The average values are listed.

In ARAKE, each node needs to consume about 16 mJ to set up a session, while 37 mJ is needed in ARAN. The nodes that have more neighbor nodes consume more energy, and those nodes in the edge of network usually expend less energy. The maximum energy cost in ARAKE is about 30 mJ, while the minimum value is only about 5 mJ.

7. Conclusions

In this paper, a novel secure routing protocol, ARAKE, for ad hoc networks is proposed to solve the anonymous secure routing problem in hostile environment. The protocol hides the topology structure of the network and can prevent most existing attacks.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This paper is supported by the Strategic Priority Research Program of CAS under Grant no. XDA06010701, the National 973 Program of China under Grant no. 2011CB302400, and the National Natural Science Foundation of China under Grant no. 61379139.

References

[1] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '99)*, pp. 90–100, New Orleans, La, USA, February 1999.

[2] D. B. Johnson, "Routing in ad hoc networks of mobile hosts," in *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, pp. 158–163, December 1994.

[3] A. Back, U. Möller, and A. Stiglic, "Traffic analysis attacks and trade-offs in anonymity providing systems," in *Proceedings of the 4th International Workshop on Information Hiding*, pp. 245–257, Springer, 2001.

[4] C. I. Fan, P. H. Ho, and R. H. Hsu, "Provably secure nested one-time secret mechanisms for fast mutual authentication and key exchange in mobile communications," *IEEE/ACM Transactions on Networking*, vol. 18, no. 3, pp. 996–1009, 2010.

[5] C. T. Li and M. S. Hwang, "A lightweight anonymous routing protocol without public key en/decryptions for wireless ad hoc networks," *Information Sciences*, vol. 181, no. 23, pp. 5333–5347, 2011.

[6] A. Boukerche, K. El-Khatib, L. Xu, and L. Korba, "SDAR: a secure distributed anonymous routing protocol for wireless and mobile ad hoc networks," in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN '04)*, pp. 618–624, November 2004.

[7] S. Seys and B. Preneel, "ARM: anonymous routing protocol for mobile ad hoc networks," in *Proceedings of the 20th International Conference on Advanced Information Networking and Applications*, pp. 133–137, Vienna, Austria, April 2006.

[8] M. G. Zapata, "Secure ad hoc on demand distance vector routing," *Mobile Computing and Communication Review*, vol. 6, no. 3, pp. 106–107, 2002.

[9] K. Sanzgiri, D. LaFlamme, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer, "Authenticated routing for ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 3, pp. 598–610, 2005.

[10] B. Zhu, Z. Wan, M. S. Kankanhalli, F. Bao, and R. H. Deng, "Anonymous secure routing in mobile ad-hoc networks," in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN '04)*, pp. 102–108, November 2004.

[11] Y. C. Zhang, W. Liu, W. J. Lou, and Y. G. Fang, "MASK: anonymous on-demand routing in mobile Ad Hoc networks," *IEEE Transactions on Wireless Communications*, vol. 5, no. 9, pp. 2376–2385, 2006.

[12] H. Shen and L. Zhao, "Alert: an anonymous location-based efficient routing protocol in manets," *IEEE Transactions on Mobile Computing*, vol. 11, no. 1, pp. 1–4, 2012.

[13] G. M. Yang, D. S. Wong, and X. T. Deng, "Anonymous and authenticated key exchange for roaming networks," *IEEE Transactions on Wireless Communications*, vol. 6, no. 9, pp. 3461–3472, 2007.

[14] J. Kong and X. Hong, "ANODR: anonymous on demand routing with untraceable routes for mobile ad-hoc networks," in *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '03)*, pp. 291–302, June 2003.

[15] S. M. M. Rahman, A. Inomata, T. Okamoto, M. Mambo, and E. Okamoto, "Anonymous secure communication in wireless mobile ad-hoc networks," in *Ubiquitous Convergence Technology*, vol. 4412 of *Lecture Notes in Computer Science*, pp. 140–149, Springer, New York, NY, USA, 2007.

[16] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[17] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology—CRYPTO 2001*, vol. 2139, pp. 213–229, Springer, New York, NY, USA, 2001.