A Secure Routing Protocol Against Byzantine Attacks for MANETs in Adversarial Environments

Ming Yu, *Senior Member, IEEE*, Mengchu Zhou, *Fellow, IEEE*, and Wei Su, *Senior Member, IEEE*

***Abstract*—To secure a mobile ad hoc network (MANET) in adversarial environments, a particularly challenging problem is how to feasibly detect and defend possible attacks on routing protocols, particularly internal attacks, such as a Byzantine attack. In this paper, we propose a novel algorithm that detects internal attacks by using both message and route redundancy during route discovery. The route-discovery messages are protected by pairwise secret keys between a source and destination and some intermediate nodes along a route established by using public key cryptographic mechanisms. We also propose an optimal routing algorithm with routing metric combining both requirements on a node's trustworthiness and performance. A node builds up the trustworthiness on its neighboring nodes based on its observations on the behaviors of the neighbor nodes. Both of the proposed algorithms can be integrated into existing routing protocols for MANETs, such as ad hoc on-demand distance vector routing (AODV) and dynamic source routing (DSR). As an example, we present such an integrated protocol called secure routing against collusion (SRAC), in which a node makes a routing decision based on its trust of its neighboring nodes and the performance provided by them. The simulation results have demonstrated the significant advantages of the proposed attack detection and routing algorithm over some known protocols.**

***Index Terms*—Ad hoc network, mobile, routing protocol, security.**

I. Introduction

THERE is an increasing need to develop and deploy highly secure mobile ad hoc networks (MANETs), particularly for military tactical and other security-sensitive operations in adversarial environments. Since a MANET does not rely on a fixed infrastructure, and network elements are wireless mobile nodes, they can rapidly be deployed with relatively low cost.

The main challenges in assuring MANET networks are due to the fact that a mobile link is susceptible to attacks, and node mobility renders the networks to having a highly dynamic topology. The attacks against routing protocols can be categorized into external and internal attacks [1]. An external attack originates from a router that does not participate in the routing process but masquerades as a trusted router. They can either advertise false routing information or generate floods of spurious service requests, such as a denial of service (DOS) attack. An internal attack originates from a compromised, misconfigured, faulty, or even malicious router inside a network domain. Among the internal attacks, *Byzantine attacks* can be defined as attacks against routing protocols, in which two or more routers collude to drop, fabricate, modify, or misroute packets in an attempt to disrupt the routing services.

Under the framework of network security, security solutions can be provided in different layers of the Open Systems Interconnection (OSI) network model. In the network layer, it is critical to provide secure routing protocols that can defend the most possible attacks against routing, which are data and routing information tampering [2]. Ad hoc routing protocols must be integrated into authentication architectures, such as public key infrastructure (PKI) and certificate authority (CA), to achieve the security requirements including confidentiality, integrity, authentication, and nonrepudiation services [3].

First, how to detect and defend internal attacks against routing protocols, such as Byzantine attacks, have been a particularly challenging problem. The problem has often been avoided by most secure routing protocols by assuming that the nodes should be trusted once authenticated. This is, unfortunately, not the case for real-world environments.

Second, what kind of authentication and key management schemes are needed to dynamically maintain a trustworthy topology and defend against malicious attacks? The security measures in mobile telecommunication networks can rely on a CA or ID-based cryptosystem [4]. However, a MANET cannot use such a CA server.

Third, the existing practice in developing secure routing protocols is by first establishing a PKI and then using cryptographic primitives to protect the messages exchanged in the routing protocols. The security and routing mechanisms are separately designed to meet the conflicting requirements: security requires using intensive computations, whereas routing needs to be efficient to properly scale [1]. Thus, the resulting protocols may be secure but not feasible or *vice versa*.

Fourth, how to quantify the engineering tradeoffs between the security and performance requirements? The problem has not well been investigated.

This paper proposes a novel attack detection and defense algorithm to solve the preceding problems for MANETs. It also develops a secure routing protocol called secure routing against collusion (SRAC) to defend Byzantine attacks as well as other internal attacks against routing protocols for MANETs in adversarial environments.

Manuscript received September 8, 2007; revised January 31, 2008 and March 15, 2008. First published April 18, 2008; current version published January 16, 2009. This paper was presented in part at the IEEE International Conference on Networking (ICON), Kuala Lumpur, Malaysia, November 2005. The review of this paper was coordinated by Dr. J. Misic.

M. Yu is with the Department of Electrical and Computer Engineering, Florida State University, Tallahassee, FL 32310-6046 USA (e-mail: mingyu@eng.fsu.edu).

M. Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: zhou@njit.edu).

W. Su is with the U.S. Army Communication Electronics Research Development and Engineering Center, Fort Monmouth, NJ 07703 USA (e-mail: wei.su@us.army.mil).

Digital Object Identifier 10.1109/TVT.2008.923683

This paper is organized as follows. Related work is reviewed in Section II. A dynamic key management scheme and an attack detection algorithm are proposed in Section III. The trustworthiness buildup scheme and optimal routing are developed in Section IV. The simulation results are presented in Section V. Section VI concludes this paper.

II. Related Work

The current secure routing protocols for MANETs can roughly be divided into two categories, i.e., 1) those adding security mechanisms to the existing routing protocols and 2) those designed to detect and defend specific attacks.

In the first category, the common practice is to secure the popular on-demand routing protocols, such as ad hoc on-demand distance vector routing (AODV) [5], destination-sequenced distance vector (DSDV) [6], and dynamic source routing (DSR) [7], by using a security association between the source and destination nodes such as pairwise secret keys and end-to-end authentication [4]. The resulting secure protocols include Secure AODV (SAODV) [8], Ariadne [9], Secure Efficient Ad hoc Distance (SEAD) [10], and Authenticated Routing for Ad hoc Networks (ARAN) [11].

SAODV is a direct extension of AODV that uses a digital signature to sign routing messages and hash chains to secure hopcounts [8], which is expensive for MANETs. Ariadne with Timed Efficient Stream Loss-tolerant Authentication (TESLA) can be considered as an extension of DSR with added security features to prevent attackers from tampering routing information and some other types of attacks such as DOS [9]. TESLA is an efficient broadcast authentication scheme, but it requires some extent of time synchronization among the nodes in a MANET [15]. SEAD is based on DSDV and uses one-way hash chains to authenticate hopcounts and sequence numbers of routing messages [10]. The security mechanism in SEAD can be TESLA or the shared secret keys between each pair of nodes. ARAN uses a digital signature to provide end-to-end authentication and provides node authentication, message integrity, and nonrepudiation services [11]. During route discovery, each routing message is signed by a source node and then broadcast to others. An intermediate node that forwards the message removes its previous hop's certificate and signature, and then attaches its own certificate and signature. During route setup, each message is similarly signed twice and unicast back to its source. Due to the use of a double signature, ARAN can defend from the most common attacks. As an authenticated routing protocol, ARAN can work with both AODV and DSR.

In the second category, protecting routing traffic against specific attacks is their major purpose. These include Network layer Protocol with Byzantine Robustness [18] for Byzantine failure and its extension to large data networks using hierarchical routing [19], ON-Demand Secure Byzantine Routing (ODSBR) [20], and Highly Secure and Efficient Routing (HSER) [21] for Byzantine attacks, Rushing Attack Prevention (RAP) [16] for rushing attacks, Secure Routing Protocol [14] for impersonation and replay attacks, and Leap-Frog [22] for a single compromised node within two hops. In ODSBR, a probing technique using binary search is proposed to find out faulty links on a path and thus to detect Byzantine behaviors. The accumulated path is then protected by aggregated signatures [23], which are unfortunately more expensive than the Rivest Shamir Adleman (RSA) signatures. In HSER, each packet is authenticated at each node by using Medium Access Control (MAC) protocols based on pairwise secret keys to detect and defend Byzantine adversaries. This approach has to be improved due to the prohibitively high computation requirement at each node. Note that ARAN is capable of defeating many identified attacks but not the Byzantine attacks discussed in this paper.

In summary, only a few protocols are capable of detecting internal attacks such as Byzantine attacks by using expensive aggregated signatures [23] or per packet filtering [21]. In addition, without a dynamic key management mechanism, the protocols could eventually fail after enough nodes are compromised.

III. Dynamic Key Management Scheme and Attack Detection Algorithm

Without losing generality, we assume that a network is equipped with several security mechanisms in different layers in addition to the network layer. For example, the application layer can have some effective intrusion detection systems to monitor anomaly behaviors that can be used to detect and defend attacks such as DOS.

In the network layer, the most possible attacks are data and routing information tampering [2]. The majority of external attacks against routing protocols can be prevented by simple link layer encryption and authentication. We propose to have every node share a unique symmetric key with the source if it needs to transmit data. By applying this mechanism, the Sybil attack, the majority of selective forwarding and sinkhole attacks, and the HELLO flood attacks can be prevented [2]. The major classes of attacks not countered are internal attacks and wormhole attacks. The defense mechanism for wormhole attacks can be found in [17]. Therefore, we focus on internal attacks that are caused by authenticated routers, such as Byzantine attacks.

A. Dynamic Key Management Scheme

There are two basic key management approaches, i.e., public and secret key-based schemes. The public key-based scheme uses a pair of public/private keys and an asymmetric algorithm such as RSA to establish session keys and authenticate nodes. In the latter scheme, a secret key is a symmetric key shared by two nodes, which is used to verify the data integrity.

Although a public key management system can be fully self-organized, the initial trust among the nodes in a network is still built by using external mechanisms. For example, Capkun *et al.* propose such a system by constructing a local certificate repository (CR) for each node [24]. The initial construction starts by issuing public key certificates based on a users' own knowledge about other users' public keys. Initially, there is a PKI or CA to distribute the knowledge among users. Therefore, the work in [24] is a dynamic maintenance mechanism in building up the certificates. Clearly, we need to assume that there are some kind of initial trusts among the nodes. For example, it is usually assumed that there exist pairwise shared secret keys among the nodes.

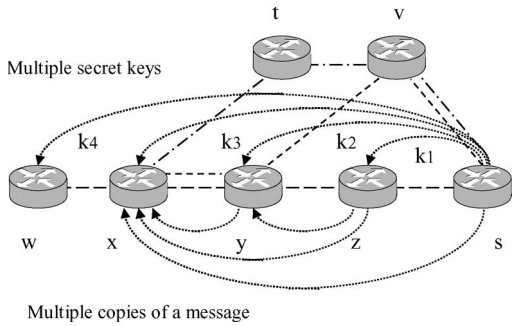

Fig. 1. Demonstration of message and route redundancy. Multiple secret keys are shared between a source and the intermediate nodes and the destination node. Multiple copies of a message are received at a destination node via different routes.

There are several methods to set up the shared keys: 1) Bootstrap the shared keys from a PKI, which might be a strong assumption for MANETs; 2) use a key distribution center, which has a shared key with each node, to build up a shared key between two nodes by using the Kerberos protocol; or 3) embed the shared keys in each node during its initialization before deployment. The third method is more practical for many MANET applications. These shared keys are then used to bootstrap the public key management system, which can create and distribute a pair of public/private keys for each node. In this paper, we assume that each node has a unique ID or address and an initial pair of public/private keys, which can be embedded into each node at the initialization of the network, or created by a self-organized public key management system [24].

We first define a network, as shown in Fig. 1, and then describe a framework of dynamic key management. Let $G = (V; E)$ be a network whose vertices in V are nodes and whose edges in E are direct wireless links among nodes. We define for each node x the set $N_1(x)$, which contains the vertices in the network G that are hop-1 or direct neighbors of x, i.e.,

$$N_1(x) = \{y : (x; y) \in E \text{ and } y \neq x\}. \quad (1)$$

Similarly, we define the hop-2 neighbors of a node as follows. For each node x, $N_2(x)$ contains the vertices in the network G that are hop-2 neighbors of x, which include neither vertices in $N_1(x)$ nor x itself, i.e.,

$$N_2(x) = \{z : (y; z) \in E \text{ and } y \in N_1(x), z \neq x\}. \quad (2)$$

Similarly, we can define the hop-n neighbors of x $[N_n(x)]$ in terms of $N_{n-1}(x)$ if the flooding path from the source to destination has n links.

As in the existing secure routing protocols, the initial trust among the nodes is built into the network by using some external mechanisms. After that, unlike the existing secure routing protocols, our framework allows a node to build up its trust on its neighboring nodes based on its observations of their behaviors. Here, important behavior is whether a node correctly routes and forwards a message to its neighbors.

Initially, a node x has a public key $K_{x,\mathrm{pub}}$ that is distributed to $N_1(x)$ by using PKI or CA. Similarly, a node y has public key $K_{y,\mathrm{pub}}$ distributed to $N_1(y)$. Thus, for example, if $y \in N_1(x)$ and $x \in N_1(y)$, i.e., x and y are hop-1 neighbors, then x can authenticate y by issuing a certificate (which is a proof of y's ID and public key with x's signature) that is signed by x with x's private key. Those who hold x's public key can now read the certificate and trust the binding of y and its public key. Based on the available certificate and key information, two hop-1 neighboring nodes can easily establish a secret key between them by using methods such as a three-way handshake [31].

Our framework for dynamic key management can be summarized as follows.

1) A secret key is established between the source and destination and some intermediate nodes along the route by using current public key information (see Section III-B).
2) Each node along the route finds out which of its direct neighbors are faulty or compromised by using the established multiple keys between the source and intermediate nodes (see Section III-C).
3) Each node updates its trustworthiness on each of its neighbors by using the observed node behavior and attack-detection results (see Section IV-A).
4) Each node constructs a local CR for the nodes it trusts. The certificates for those compromised nodes are immediately revoked. A node may expand its CR by adding newly trusted nodes or exchanging repository information with its trusted neighbors.
5) By combining the current CR information and existing maintenance procedures for public key management, the nodes in the network can update public key information or build up a self-organized PKI, as in [24].

In this paper, we focus the first three steps in the following sections.

B. Key Distribution and Node Authentication

We define the notations as follows. s denotes the sender node; r denotes the receiver node; $K_{s,\mathrm{pub}}$ and $K_{s,\mathrm{pri}}$ denote the public and private keys of node s, respectively; $E(m, K)$ denotes the public key encryption algorithm with a key K on message m, where $m = M + \{ID_f\} + S_N$, and M is the original message; ID_f denotes the ID of f, which is the node that forwards the message m; S_N is the sequence number of the message; and $h(m + k)$ denotes the keyed hash algorithm with a key k on message m, where $+$ denotes the concatenation of strings. It can be seen that any node that handles the message has to append its ID for nonrepudiation service. The ID is protected together with the forwarded message.

Whenever there is a need for a node to initiate a route discovery process, it creates pairwise shared keys with intermediate nodes, hop by hop, until it reaches the destination. First, it picks random number num. Then, it signs num with its private key by using a public key algorithm like RSA. After that, the route discovery message is protected by a keyed hash MAC algorithm such as MD5 [31]. Finally, the hash value and signature can now be attached to the route discovery message and sent out to its neighbors. The complete route request (RREQ) packet sent by the node can be summarized as

$$m + h(m + num) + E(num, K_{s,\mathrm{pri}}). \quad (3)$$

Those who are s's neighbors and have its public key are able to verify the signature and thus decrypt the key in the message.

Suppose that $z \in N_1(s)$ is one of s's hop-1 neighbors. Whenever there is a need for s to initiate a route discovery process, it picks a key k_1 at random, which will serve as the shared secret key between s and z. Then, s encrypts the key k_1 by using its neighbor's public key $K_{z,\mathrm{pub}}$. After that, it encrypts the above encrypted key by using its own private key $K_{s,\mathrm{pri}}$. The result serves as a signature for the route discovery message, which is protected by a keyed hash MAC algorithm such as MD5. The complete procedure is called Keyed MD5 [31]. The complete RREQ sent by s can be summarized as

$$m_q + h(m_q + k_1) + E\left(E(k_1, K_{z,\mathrm{pub}}), K_{s,\mathrm{pri}}\right), \quad \text{for } z \in N_1(s) \tag{4}$$

where m_q stands for the message used in RREQ. This way, only the node that has z's private key can read the key k_1, the receiving node is also assured that the key and message come from s, and finally, the integrity of message m can be verified by the receiving node after it decrypts the key. Then, z sends back s a route reply (RREP) packet in a similar format

$$m_p + h(m_p + k_1) + E\left(E(k_1, K_{s,\mathrm{pub}}), K_{z,\mathrm{pri}}\right), \quad \text{for } z \in N_1(s) \tag{5}$$

where m_p stands for the message used in RREP. By decrypting the message and comparing the key, s can authenticate z and distribute a shared key to z. Similarly, s establishes a shared key with each of its hop-1 neighbors.

Suppose that $y \in N_1(z)$. z can also similarly find out its hop-1 neighbors and also establishes a shared key with each of them. For s to send messages to its hop-2 neighbors, i.e., $N_2(s)$, for example, y, s requests z to forward the message to y. In z's handshaking with y, z can pick s's public key instead of a random key and send it to y. This way, s's public key can be delivered to its hop-2 neighbors. Similarly, s can obtain the public keys of its hop-2 neighbors.

By checking the acknowledgement message back from y via z, s can find out all of its hop-2 neighbors $N_2(s)$. Therefore, s can send a message to $r \in N_2(s)$ via $z \in N_1(s)$ in the following format:

$$m_2 + h(m_2 + k_1),\ k_1 =: \text{shared key between } s \text{ and } y \tag{6}$$

where

$$m_2 = m + h(m + k_2) + E\left(E(k_2, K_{r,\mathrm{pub}}), K_{s,\mathrm{pri}}\right) \quad \text{for} \quad r \in N_2(s) \tag{7}$$

where k_2 is the shared key between s and its hop-2 neighbor r. Similarly, by using the double hash and signature operations, the shared key between s and its hop-n neighbors, i.e., k_n, is created by s and distributed to $N_n(s)$, where $n = 2, 3, \ldots$.

In the above key distribution process, the same message m has been sent to the destination multiple times and protected by different secret keys at each time. This is what we call *message redundancy*. To utilize the message redundancy, the implementation is simple: each node is required to receive multiple copies of the same route discovery message before sending back an acknowledgement.

It is noted that receiving multiple copies, instead of the first copy, incurs overhead to the route discovery process. The number of copies is determined by two factors. The first one is security, i.e., the trustworthiness of the nodes in the network. To build a route with a certain amount of trustworthiness, the destination needs to evaluate more copies in a less-trusted environment than in a more-trusted one. The second one is performance, i.e., the timeout value of the route request message. Receiving more copies means a larger value of the timeout and thus results in a higher routing latency. Therefore, an optimal value of the number of the copies can be found by trading off the two factors, which are application specific. In our simulations, the number of copies is found to be in the range of 2–4 for the cases in which 10%–40% of the nodes are malicious.

It is worth pointing out that the use of digital signatures and message redundancy may restrict the applications of the proposed key management schemes to routing protocols, as we will analyze in Section IV-D.

C. Route Discovery and Attack Detection

Based on the key management mechanism, the next task is to develop a framework for the secure discovery of the dynamic network topology. The attack detection scheme is incorporated into topology discovery procedures.

Route discovery is straightforward for a node after it decrypts the received route discovery messages. To discover the routes in a dynamic environment, we need to use the inherent redundancies of the routes in ad hoc networks, called *route redundancy*, which means there are multiple, possibly disjoint, routes between nodes. As long as there are sufficiently many correct nodes, the routing protocols should be able to discover routes that go around some compromised nodes. Many ad hoc routing protocols such as AODV and DSR can discover multiple routes. Similar methods can be adopted into our scheme to discover multiple routes.

Once the security associations between a source and destination have been established, and trustworthy routes have been identified from source to destination, the source can simply use the shared key to protect the data traffic sent to the destination: $m + h(m + k_{sd})$, where k_{sd} is the key shared between the source node (s) and destination node (d).

To detect internal attacks, including Byzantine attacks, we assume the following.

1) Each node has a pair of public/private keys and a unique ID. A compromised node participates in routing until detected.
2) The source and destination nodes are secured by external security agents. There is a shared key between the source and destination nodes.
3) Each of the intermediate nodes between the source and destination has established a shared key with the source node by using the key management scheme described in Section III-B.
4) There are enough uncompromised nodes in the network so that a message can arrive at the destination via different routes.

In this section, we extend our algorithm in detecting collusion to Byzantine attacks, in which two or more nodes collude to drop, fabricate, modify, or misroute packets, and these nodes are consecutively located on a path [30].

1) Detection of a Single Malicious Node: The basic mechanism for a node to detect misbehaving nodes is to compare the different copies of the same message it has received via different routes or at different times. The nodes along a route can be found out by checking the aggregated node IDs that are attached to the message. When a message comes from different intermediate nodes, it has to be decrypted by using different shared keys.

To be more specific, we assume that z (in Fig. 1) is a compromised node during the route discovery phase, although it is initially authenticated. Clearly, z could not tamper the message from s to y because the message is protected with a key between s and y. Of course, z may simply drop the message when it needs to forward the message to y. However, there are at least two copies of the same message y expects to receive. By comparing these copies from other neighbors, y is still able to detect that z is faulty or compromised.

Similarly, y can also detect other internal attacks, such as message fabrication caused by z. Therefore, the attacks initiated by a single inside node can be detected.

2) Detection of Two Colluding Nodes: A more challenging case is the Byzantine attack. In our design of key management schemes, a source has directly established a shared key with each of its hop-n neighbors.

Suppose that both z and y are compromised and colluding. In addition, s shares a hop-1 key with z (i.e., $k_{1,sz}$), a hop-2 key with y (i.e., $k_{2,sy}$), and a hop-3 key with x (i.e., $k_{3,sx}$). During route discovery, x may receive three copies of a message m from s and via different intermediate nodes y and z, respectively, in the following formats:

$$\begin{aligned} C_1 &= m + h(m + k_{3,sx}) \\ C_2 &= m + h(m + k_{2,sy}) + h\left(m + h(m + k_{2,sy}) + k_{1,yx}\right) \\ C_3 &= m + h(m + k_{1,sz}) + h\left(m + h(m + k_{1,sz}) + k_{1,zy}\right) \\ &\quad + h\left(m + h(m + k_{1,sz}) + k_{1,yx}\right). \end{aligned} \tag{8}$$

Suppose that z and y are two colluding nodes. It is assumed that the source and destination, i.e., s and x, are trusted via some external mechanisms. Note that each copy of the message is verified by an intermediate node along a route. As a single node, z cannot tamper the message without being detected.

Let us assume that z has modified the message but y does not tell during its forwarding. After having received the three copies from s, x finds the discrepancies among C_1, C_2, and C_3. Note that C_1 directly comes from s and thus can be trusted; y cannot change the message without being detected, and thus, C_2 must match C_1. Therefore, C_3 has been modified, and x finds that there may be some compromised or faulty nodes among the nodes that forward the message, e.g., z and/or y. It can be seen from C_3 that z may modify the message and then forwards it to y, who also gets a copy of the message directly from s as seen in C_2. If y reports the discrepancies of the two copies, then z must be a compromised node. Otherwise, both y and x are compromised and colluding nodes, although y does not change the message.

It is worth pointing out that a receiving node can select a message most likely to be right among the multiple copies by using voting algorithms or Bayesian estimation [26] or check its local CR to find out who is more trustworthy among its neighboring nodes that forward the same message. If all the neighboring nodes are equally trustworthy, the receiving node can simply choose one of the copies. It can also choose a copy that comes in a route with better performance, as we will see in Section IV-B.

3) Detection of More Colluding Nodes: Similarly, for the case of three colluding nodes consecutively located on a route, their collusion can also be detected if there exists at least four copies of the message that arrives at the receiver.

Generally, to not only detect the collusion of n compromised nodes that are consecutively located on a route but also identify these nodes, a receiver must have at least $n+1$ copies of the same message, and one of the copies is more trusted than the others. The copies can either go through different routes or be protected by the shared keys in different segments of a route.

In summary, the internal attacks initiated by a single compromised node and the Byzantine attacks can be detected without using expensive aggregated signatures, which are used to protect a route from end to end.

It is also noted that the trustworthiness of the source node can be solved only via external mechanisms such as PKI by using such mechanisms as key refreshing, rekeying, and revoking. We also note that the redundant use of shared keys between a source and each intermediate and the destination node may result in a scalability problem. For example, if there are n nodes along a route, then the dynamic key management scheme needs to create and distribute $(n-1)^2/2$ keys to the nodes on the route. Therefore, it is not appropriate for networks with a large number of low-resource nodes.

IV. Trust Modeling and Optimal Routing

A. Trust Modeling and Evaluation

We define the trustworthiness on a node n by another node x as the probability that n will perform a particular action expected by x, which is denoted as $T_x(n)$, irrespective of the ability to monitor or control n. The trustworthiness can be evaluated by x in terms of its knowledge accumulated during a specific operation period by using weighting average over the trust on each category of actions, including route request, route reply, route error, and data transmission [27].

We assume that during an observation period, x has received a total of m_t message transmissions from n, among which m_c's are found to be correct; the total number of attempted transmissions is m_a; and the total number of successful transmissions is m_s. Then

$$T_x(n) = \frac{m_c + \epsilon m_s}{m_t + \epsilon m_a} \tag{9}$$

where $0 < \epsilon < 1$ is a weighting factor that represents a ratio of the successful transmissions, which reflects the probability that the link correctly works. Here, we adopt a statistical model

similar to the one used for measuring link quality in [25], which is different from the trust level evaluation in [27]. The model in (9) not only evaluates the trustworthiness but also partially reflects the link quality. Other more complicated measurements used to model the link quality, such as the collision detection and signal separation technique [29], and link adaptation and power control algorithm [13] may also be applied to obtain a more accurate trustworthiness value.

Denote by $T_x(n;j)$ the trustworthiness in node n, which is assigned by node x during the jth trustworthiness updating cycle. Every time a new observation comes in, the node updates its repository and calculates a trustworthiness value by using a weighted average or moving average model. Assume that during the jth trustworthiness updating cycle the measurement of $T_x(n;j)$ is denoted as $\tilde{T}_x(n;j)$, which is computed based on n's current behavior when x checks the correctness and validity of the messages that come from n. During the $(j+1)$th trustworthiness updating cycle, these values are used to obtain an estimate of the trustworthiness, which is denoted as $\hat{T}_x(n;j)$.

To obtain a smooth estimation, we use a moving average model

$$\hat{T}_x(n;j+1) = \alpha\hat{T}_x(n;j)+(1-\alpha)\tilde{T}_x(n;j), \quad \text{for } n \in N_1(x) \tag{10}$$

where $0 < \alpha < 1$ is a weighting factor used to tradeoff between current measurement value and previous estimate.

Consider a path $p \in P_{s\to x}$, where $P_{s\to x}$ is the set of paths that start from a source node s to a destination node x, i.e., $P_{s\to x} = \{\text{all paths from } s \text{ to } x\}$. Denote by $T_x(p;j)$ the trustworthiness of the path assigned by node x. Thus, the path trustworthiness can be expressed as

$$T_x(p;j) = \prod_{n\in p} T_x(n;j). \tag{11}$$

If y is on the route from s to x, i.e., $y \in p$, then $y \in N_1(x)$. Denote by p_1 the path from s to y. Then, we have

$$T_x(p;j) = T_x(y;j) \prod_{n\in p_1} T_y(n;j) = T_x(y;j)T_y(p_1;j). \tag{12}$$

Therefore, x can build up its trustworthiness on a path based on its trustworthiness on its neighboring nodes. The relationship in (12) is also used as a routing metric for a node to make routing decisions.

B. Mathematical Formulation of Optimal Routing

The routing metrics can be quantified as follows. First, we need to consider the trustworthiness of each candidate route. We assume that each node has locally built up a trustworthiness repository for the nodes it knows based on its CR and current behavior observed in the topology discovery phase. A destination node has also calculated a value of trustworthiness for each possible route from the source node, as shown in Section IV-A. The repository is updated every time the topology is rediscovered.

Second, the performance requirement must be considered in making routing decisions.

Assume that the transmission capacity of the wireless link that originates from a node n is B_n. The traffic requirement for the link is F_n, which is measured in the same units as B_n. For delay-sensitive traffic, we also use τ_n to represent the processing and propagation delay when being delivered by n to its next hop. A frequently used objective function is [32]

$$Q_x(p) = \sum_{n\in p}\left(\frac{F_n}{B_n - F_n} + \tau_n F_n\right), \quad \text{for } p \in P_{s\to x} \tag{13}$$

which is the average number of packets in the network based on the hypothesis that each queue behaves as an $M/M/1$ queue of packets. Note that for a link on path p, a smaller value of $Q_x(p)$ is preferred, either because of a smaller delay or a relatively larger link capacity. Rewriting (13) into a recursive format, we have

$$Q_x(p) = Q_y(p_1) + \frac{F_x}{B_x - F_x} + \tau_x F_n, \quad y \in N_1(x), \quad y \in p_1 \in p. \tag{14}$$

Combining both requirements on the trustworthiness and performance, a path that is less trustworthier and does not meet the desired performance must be penalized in our objective; thus, a combined cost function can be designed as

$$D(p) = \beta\left(1 - T_x(p;j)\right) Q_x(p), \quad \text{for } p \in P_{s\to x} \tag{15}$$

where $\beta > 0$ is a constant used to scale the value of the cost function.

Now the routing problem can be written as

$$\text{minimize } D(p), \quad \text{for } p \in P_{s\to x} \tag{16}$$

and subject to the constraints

$$B_n - F_n \geq 0; \quad \tau_n \geq 0; \quad T_{\min} \leq T_x(n;j) \leq 1; \quad \text{for } n \in p \tag{17}$$

where $T_{\min}$ is the minimum trustworthiness required for a node to be allowed to join in a route. In (15)–(17), we explicitly integrate the security and performance requirements into a routing problem. Note that if the intermediate nodes between s and x have the same levels of trustworthiness, then all the routes between s and x can equivalently be measured by the traditional hopcounts. The routes of the same hopcounts will have the same levels of trustworthiness. In this case, the optimal route is only determined by the performance requirements, as shown in (15), provided that the requirement on the minimum trustworthiness is met. By solving the optimization problem, we can develop a routing algorithm.

Many algorithms in distributed routing can be used to find a global optimal solution, e.g., the Dijkstra algorithm and the distributed asynchronous Bellman–Ford algorithm [32]. Alternatively, we can use the heuristic QoS routing algorithm [12] by applying the objective function in (15). By using the latter, we may be able to find a solution with the minimum $D(p)$, but the constraints in (17) may not be met.

To avoid intensive computation and communication and ensure good scalability, we develop a routing algorithm to solve

the problem while guaranteeing local optimality. The source node selects multiple routes as candidates. Each intermediate node along the candidate routes computes a cost from the source and passes it the next node on the way to the destination. The cost contains two parts, i.e., T_x and Q_x, in terms of (12) and (14), respectively, which can recursively be calculated based on the nodes along the route. Therefore, each route can be assigned an index called Trustworthiness–QoS index (TQI), i.e., a number to represent the combined trustworthiness and performance cost along the route, by each intermediate node along the route. The destination chooses a final route among the candidates in terms of the accumulated TQI value. It is the responsibility of each intermediate node to make a choice of its best interest in weighting T_x and Q_x.

C. Routing Algorithm

The heuristic algorithm can be summarized as follows.

1) During route discovery, a source node sends RREQ packets to its neighboring nodes. In these packets, along with the regular information, the node also sends its security-related information, such as key information, as outlined in Section III-B.
2) Once an RREQ packet is received by an intermediate node, it calculates the TQI by using (15). The node places the link trustworthiness and QoS information in the RREQ packet and forwards it to its next hop. This process is repeated until it reaches the final destination.
3) At the destination, the node waits for a fixed number of RREQs before it makes a decision. Or else, a particular time can be set for which the destination or intermediate node needs to wait before making a routing decision. Once the various RREQs are received, the destination node compares the various TQI index values and selects the index with the least cost. It then unicasts the RREP back to the source node. When the source node receives the RREP, it starts data communication by using the route.
4) Once the route is established, the intermediate nodes monitor the link status of the next hops in the active routes. Those that do not meet the performance and trustworthiness requirements, as shown in (17), will be eliminated from the route.
5) When a link breakage in an active route is detected, a route error (RERR) packet is used to notify the other nodes that the loss of that link has occurred. Some maintenance procedures are needed as in AODV.

Compared with the existing ad hoc and QoS routing algorithms, this algorithm for the first time intends to optimize a combined objective function of security and performance parameters. The existing work [9]–[11], however, considers security and performance in a separate or nonquantitative fashion in their routing algorithm, thereby leaving no room for tradeoff between them.

It is worth noting that the proposed algorithm is only a local optimization method, and the solution may not globally be optimal. On the positive side, the complexity of the solution to the routing problem is greatly reduced.

TABLE I
SIMULATION PARAMETERS FOR SRAC AND ARIADNE [9]

Number of nodes	50
Topology dimensions	1500m × 300m
Radio range	250m
Node pause times	0-900s
Maximum node speed	20m/s
Source-destination pairs	20
Traffic pattern	CBR/UDP
Data payload size	512 bytes/packet
Source data rate (each)	4 packets/s
Total data rate	327 kbps
Wireless link capacity	2000 kbps

D. Overhead Incurred by Secure Routing Mechanisms

To investigate the routing performance, we need to estimate the overhead on the packet processing time, i.e., the extra time needed by using encryption and decryption.

As shown in the previous sections, the routing (or control) packets, including RREQ, RREP, RERR, and HELLO packets, are encrypted. The data packets are only protected with keyed hash MAC such as MD5. To process a routing packet (e.g., RREQ), a node needs to receive an encoded message that contains the RREQ from one of its neighbors, verify that the certificate attached by the neighbor is valid, replace the certificate with its own certificate in the attachment, sign the message, conduct a hash operation on the message, and transmit the message to its next hop, as shown in (4). In summary, the total processing time by a node (denoted by T_n) includes two RSA signature generations (denoted by t_g), two RSA signature verifications (denoted by t_v), one hash operation (denoted by t_h), plus the time spent in receiving (denoted by t_r), transmission (denoted by t_t), queueing (denoted by t_q), and propagation (denoted by t_p), that is, $T_n = 2t_g + 2t_v + t_h + t_r + t_t + t_q + t_p$. For a typical network configuration, the last five terms are very small. For example, MD5 can process packets at a typical speed of 600 Mb/s, that is, for a typical message of 1 kb, it takes $t_h = 1.7408$ μs to perform the hash operation. In addition, for the values shown in Table I, we can find that $t_r = 0.5$ μs, $t_t = 0.5$ μs, $t_q = 0.0977$ μs, and $t_p = 0.8333$ μs. The five terms have a total of 3.67 μs and thus can be neglected.

It is estimated that for an RSA key with a length of L_k (in bits), the CPU cycles needed to perform one RSA operation is about $(L_k + 2) \times (L_k + 2 + 32)$ for a typical implementation [28], which is equal to 0.28 and 1.09 million for $L_k = 512$ and 1024, respectively. It is also estimated that the generation of a signature takes about 20 RSA operations, whereas the verification takes only one RSA operation. For a CPU of 2.8 GHz (e.g., an Intel Pentium 4 processor), we can find $t_g = 2.0$ and 7.8 ms, $t_v = 0.1$ and 0.39 ms, and thus, we have $T_n = 4.20$ and 16.38 ms for $L_k = 512$ and 1024, respectively. Similarly, for a CPU of 206 MHz (e.g., an Intel Strong Arm 32-bit basic processor), we can find $t_g = 27$ and 105.44 ms and $t_v = 1.35$ and 5.27 ms, and thus, we have $T_n = 56.70$ and 221.34 ms for $L_k = 512$ and 1024, respectively.

It can be seen that the major overhead is caused by the encryption and decryption algorithms, which may restrict the applications of the secure routing protocol. For MANETs with high levels of mobility, links are likely to be broken frequently,

TABLE II
SIMULATION PARAMETERS FOR SRAC AND AODV

Number of nodes	50
Topology dimensions	1000m × 1000m
Radio range	250m
Node pause times	0-40s
Maximum node speed	1-20m/s
Source-destination pairs	20
Traffic pattern	FTP/TCP
Data payload size	32-1060 bytes/packet

and thus, the protocol needs to handle many RERR packets. For example, for a CPU of 206 MHz, $L_k = 1024$, it takes $T_n = 221.34$ ms to handle one RERR packet. To send back an RREP on an RREQ, it may even take 2–4 times of T_n, e.g., 0.4427–0.8856 s, due to the use of message redundancy. Therefore, the proposed protocol is not appropriate for devices with a low computing power. Otherwise, we need to limit the size of the encryption key. For example, if we choose $L_k = 256$, we can find that $T_n = 15.25$ ms for CPUs of 206 MHz.

V. SIMULATION RESULTS

In this section, an NS-2 simulator is used to investigate the performances of SRAC and compare it to other protocols [33].

A. Network Models and Parameters

The parameters and values are given in Tables I and II for the simulations used to compare SRAC to AODV and Ariadne, respectively. The mobility of nodes is generated using a random waypoint model wherein a node starts off at a random point in the topology. The radio propagation model used is a two-ray ground reflection that accounts for a realistic physical scenario.

The simulations are conducted on a Dell PowerEdge server with two Intel Xeon processors of 2.66 GHz and 4-GB SDRAM running in a Linux OS of Fedora Core version 3.0. The encryption and decryption operations on routing packets are simulated by adding a time of $T_n = 16.38$ ms (based on a CPU of 2.8 GHz) to the processing time per packet per node, as shown in Section IV-D. The RSA key size is assumed to be 1024 bits. The encryption and decryption times are invoked whenever a node generates, receives, or forwards a routing packet, which increases the overhead of SRAC as compared to AODV. However, the total overhead has in fact been reduced in the presence of malicious nodes in the network, as shown in the following sections.

To include the behavior of malicious nodes into the simulations, SRAC is implemented by modifying the AODV protocol in NS-2. We make the following assumptions on a malicious node.

1) It does not have knowledge about the public key of its hop-1 neighbors prior to the route-discovery phase. The session keys are established on demand.
2) Once the route-discovery phase is accomplished, a malicious node randomly drops both routing and data packets or selectively only drops data packets with a specific probability, i.e., the dropping ratio.

In this simulation, as a demonstration purpose, a malicious node just drops data packets. It is worthy noting that other malicious behaviors, such as false advertising, misrouting, and violating security rules, can be detected and defended in the same way as dropping. We also assume that a source–destination pair (SD pair) is trusted and cannot turn to be malicious throughout the operation of all these protocols.

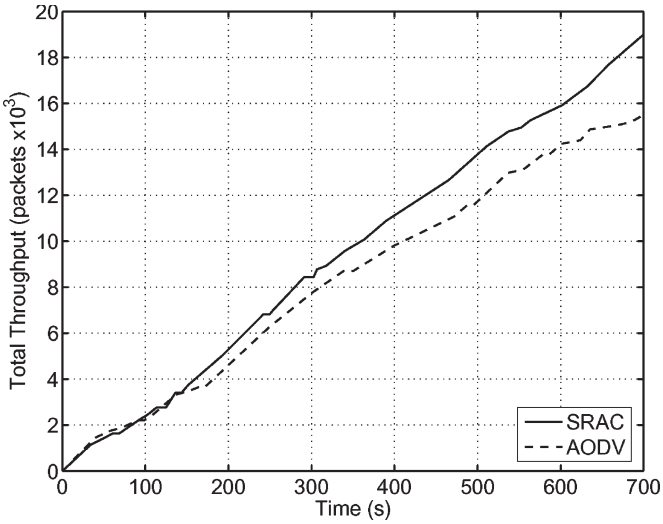

Fig. 2. Total throughput in the presence of five malicious nodes.

B. Performance Evaluation

The performance metrics are defined as follows.

1) *Total throughput*: The total number of data (application) packets that have been received at time t by a destination node.
2) *Total overhead*: The total number of routing (control) packets that have been transmitted at time t by the nodes in the network.
3) *Packet latency*: The time elapsed since a data packet is transmitted to the time when it is received at the destination.
4) *Packet delivery ratio (PDR)*: The ratio of the total number of data packets successfully delivered to the destination to the total number of data packets sent out by a source node.

In our first scenario, simulations are conducted to examine the performance impact of adding security to routing protocols. Here, SRAC is compared to AODV because the implementation of SRAC is based on AODV. In the simulations, the parameters are given in Table II. A malicious node randomly drops data packets and can be detected during topology discovery. The dropping ratio is in the range of 20%–50%. Each simulation is run for 700 s to collect the performance data.

Fig. 2 shows the total throughput of SRAC and AODV in the presence of FIVE malicious nodes out of 50. The number of malicious nodes is so low that AODV continues delivering packets, although with a less amount than SRAC. This is due to the fact that packet dropping is not serious, and there are enough uncompromised nodes available to establish routes between SD pairs.

However, if the number of malicious nodes increases from five to 10, and to 20, as shown in Figs. 3 and 4, the number of packets delivered by AODV drastically decreases, whereas

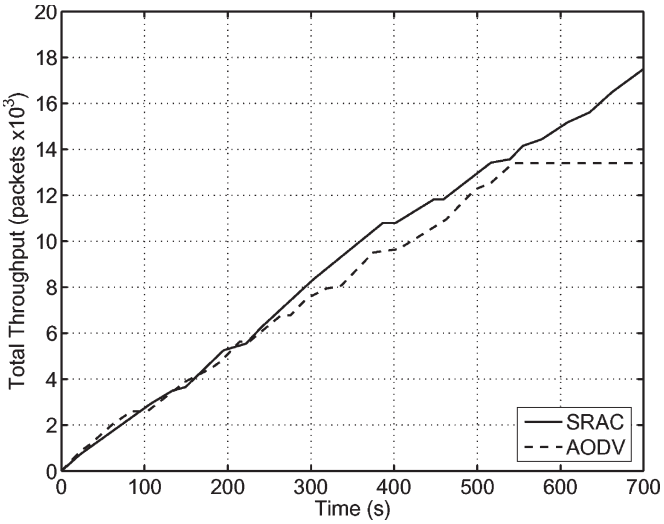

Fig. 3. Total throughput in the presence of ten malicious nodes.

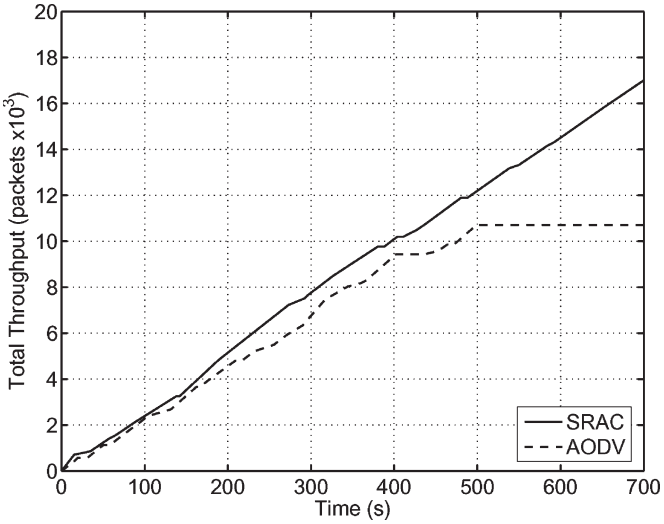

Fig. 4. Total throughput in the presence of 20 malicious nodes.

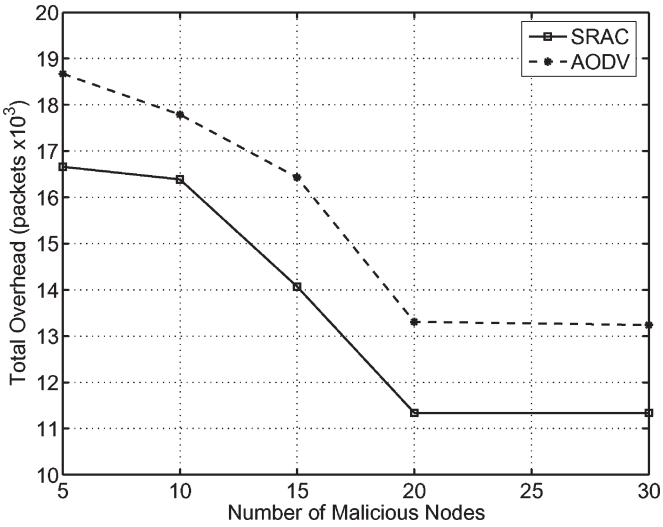

Fig. 5. Total overheard of SRAC and AODV in the presence of different numbers of malicious nodes.

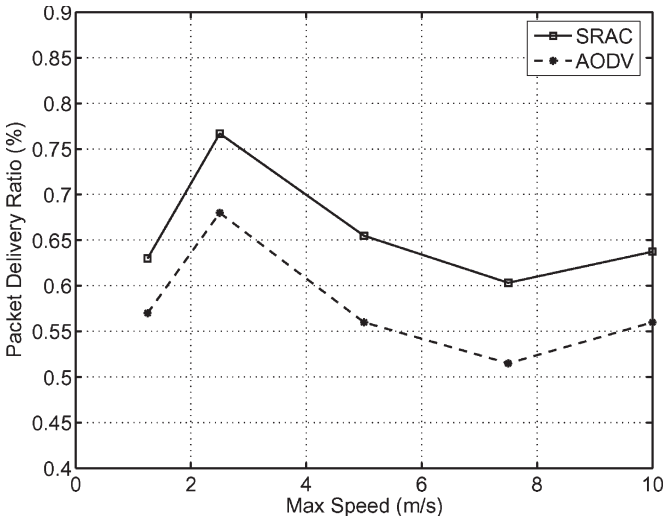

Fig. 6. Packet delivery ratio of SRAC and AODV at different speeds.

SRAC still delivers almost the same amount of data as in the previous case. The reason is that most of the routes in AODV have to go through some malicious nodes and thus result in a high packet drop or low PDR. Eventually, AODV stops delivering packets at $t = 550$ and 500 s in the presence of 20% and 40% malicious nodes, respectively. Due to the packet drop, a connection will be timed out, and a new route discovery will be reinitiated. However, with a high probability, the new routes may again contain some malicious nodes and thus result in high data loss. On the other hand, SRAC tries to discover the paths that can go around malicious nodes. Even when the number of malicious nodes is relatively high, SRAC still discovers trustworthy routes and thus assures successful packet delivery.

Fig. 5 shows the total overhead of SRAC and AODV. It can be seen that SRAC always has a smaller overhead than AODV in the presence of different numbers of malicious nodes. The reason is that SRAC can detect malicious nodes and thus exclude them from routing. For AODV, if a node on an established route becomes malicious and starts dropping packets, the source that waits for the acknowledgment (ACK) eventually times out. A new route discovery is initiated to reestablish the route. As the number of malicious nodes increases, AODV tends to wait and time out more often, thus delivering increasingly fewer routing and data packets. Therefore, the total overhead is reduced. For SRAC, since the behavior of neighboring nodes is monitored, the malicious nodes are detected and excluded from routing. As the number of nodes that join routing becomes fewer, the total overhead is reduced to a larger extent than that of AODV.

In our second scenario, it is assumed that 50% of the nodes are malicious. The parameters are shown in Table II, except that the number of nodes is 100, and the traffic is the constant bit rate with pattern in Table I. A malicious node randomly drops both routing and data packets with a dropping ratio of 80%. The maximum nodal speed varies between 1.25 and 10 m/s.

Fig. 6 shows the PDR of SRAC at different speeds as compared to AODV. It can be seen that SRAC always outperforms AODV in PDR, because SRAC always chooses more reliable routes by avoiding malicious nodes. At low levels of mobility, as the maximum speed increases from 1.25 to 2.5 m/s, although the increased link breakage may reduce the PDR, the SD pairs are more likely to find available nodes to forward the packets.

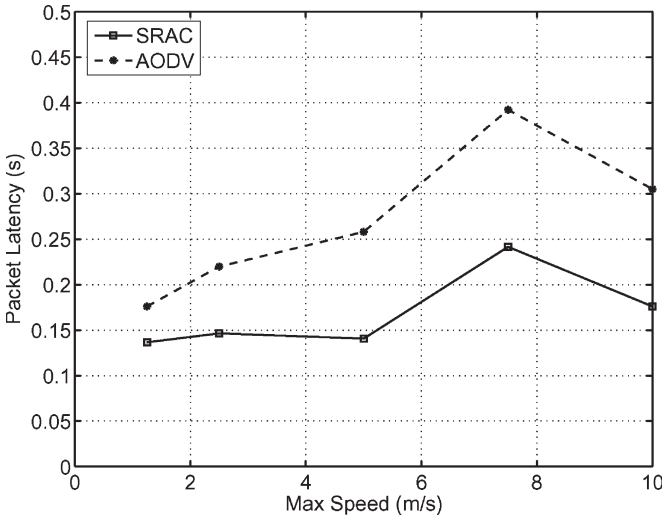

Fig. 7. Packet latency of SRAC and AODV at different speeds.

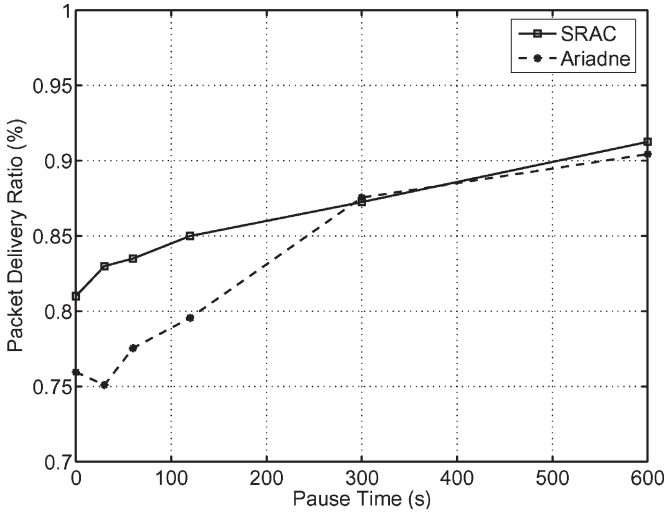

Fig. 8. Packet delivery ratio of SRAC and Ariadne at different pause times.

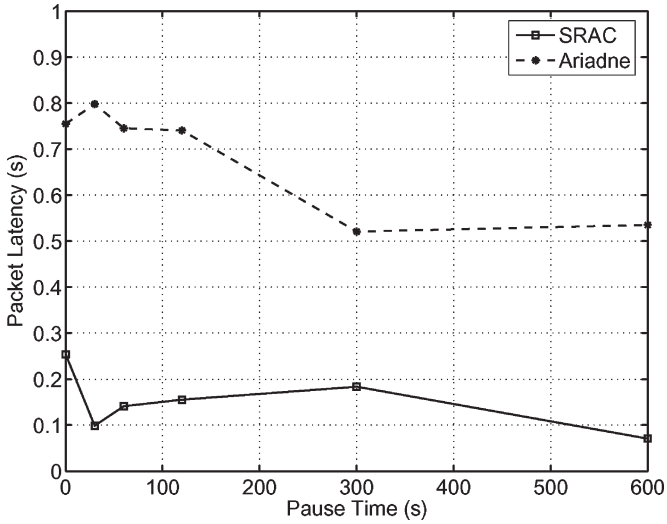

Fig. 9. Packet latency of SRAC and Ariadne at different pause times.

At high levels of mobility, as the maximum speed increases from 2.5 and 10 m/s, the link breakage becomes the major cause that reduces the PDR. The SD pairs do not have adequate time to find alternative routes due to a fixed route request timer. Therefore, the PDR decreases as the maximum speed increases.

Fig. 7 shows the packet latency of SRAC at different speeds as compared with AODV. For both SRAC and AODV, as the speed increases, the information on a route in the routing table will quickly be out of date. Thus, both take more time to establish routes. However, SRAC always has a lower packet latency than AODV. One reason is the use of multipath routing in SRAC. For AODV, its routing table only stores one path. It has to reestablish another route once a link is broken. For SRAC, its routing table contains multiple paths. If one fails, an alternate one will immediately be available. Another major reason is that the packet performance and route reliability are both considered in the routing metric, as defined in (15).

In our third scenario, SRAC is compared to Ariadne, which is also an on-demand secure routing protocol targeted for MANETs in adversarial environments. The performances of SRAC are evaluated with the parameters in Table I and compared to those of Ariadne reported in [9] at different levels of mobility. Note that a smaller pause time means a higher level of mobility and *vice versa*.

Fig. 8 shows the PDRs of both SRAC and Ariadne with different pause times. It can be seen that at low to medium levels of mobility, the PDRs of both protocols are almost the same. While at medium to high levels of mobility, SRAC has a higher PDR than Ariadne. At low levels of mobility, the impact of adding time to process security is not significant; thus, SRAC and Ariadne have similar PDRs, as in AODV and DSR. Note that SRAC is based on AODV, whereas Ariadne is based on DSR. At high levels of mobility, the performance difference in PDR is mainly due to the addition of security mechanisms. When the speed is high, links are more likely to be broken, and thus, more RERR packets need to be handled. In SRAC, the impact on a routing packet is adding an extra packet processing time, e.g., $T_n = 16.38$ ms. In Ariadne, the security mechanism using TESLA requires that a message cannot be verified until the TESLA key is disclosed, i.e., the TESLA time interval (TTI). For example, for TTI $= 1$ s as in Ariadne, if a sender receives an RERR packet caused by a broken link, it continues to send packets along the broken route until the RERR packet is verified by using a newly released hash key, which is about half of TTI, i.e., 0.5 s. SRAC does not have such a delay and thus delivers more packets to the destination, i.e., it has a higher PDR than Ariadne.

Similarly, SRAC consistently has a lower packet latency than Ariadne, as shown in Fig. 9. In addition to the above half TTI delay, another reason is that SRAC uses a routing metric that contains the trustworthiness and performance metrics of the nodes along a route. Without this routing metric, some stale and unreliable route may be selected, which has a short lifetime. Therefore, a new route needs to be discovered, which increase not only the total overhead but also the packet latency.

C. Security Analysis

In this section, we analyze the security of SRAC in the presence of different attacks. We also compare it to Ariadne

and ARAN because most of the attacks there can be defended by SRAC.

Byzantine Attacks: In SRAC, a node receives multiple copies of the same message, which are forwarded by different nodes over different routes and protected by different keys at different segments on a route. The node can find discrepancies by comparing these copies. If some of the nodes that forward the message are more trustworthier than others, it can immediately claim those that do not forward a message correctly as colluding nodes [30].

On the other hand, Ariadne and ARAN are not designed for this type of adversarial environment. For example, as shown in Fig. 1, node x is next to the destination w. It attaches its own certificate to a message and forwards the message to z via y. If x pretends to be the destination and y does not report the violation, i.e., they are colluding, then z will not find out the truth. The reason is that there is no other reference in Ariadne and ARAN that can be used to compare and validate the correctness of a message.

Unauthorized Participation: In SRAC, all the packets are encrypted with either an asymmetric or shared key. Only authorized nodes have the asymmetric key and use the key to get the shared key with each other. Only authorized nodes have both keys. Thus, unauthorized nodes are prohibited to join a route.

Spoofed Route Signaling: Route discovery packets are encrypted by a source and destination using either an asymmetric or shared key. Only the source and destination have the right keys to decrypt them. Thus, the impersonation attacks are completely prevented.

Fabricated Routing Messages: In SRAC, each routing message is verified at an intermediate node. The message reaches the destination through multiple paths. By using message and route redundancy, an intermediate node will be detected if the message is modified. Later, a lower trustworthiness value will be given to that node. When the trust value of the node is below a threshold, the node will be labeled and removed from the routing table. Therefore, the fabrication of routing messages can be prevented. Note that ARAN does not have the mechanism to prevent such attacks.

Securing Shortest Paths: In SRAC, multiple paths are built from a source to a destination during route discovery. The destination can choose a path with minimum cost to send back a reply, which can be the most trusted or shortest path with minimum delay, the shortest path with minimum hopcounts, or a combined cost, as in (15). Whereas ARAN may choose a path as the shortest path on which an RREQ packet arrives at the destination first. Note that the total time of the RREQ experience consists of two parts: 1) the queueing, processing, and propagation delay and 2) the security processing time on the packet. In some cases, the second part can be much longer than the first. Thus, ARAN may actually choose a longer path.

Key Distribution Attack: In SRAC, if node x creates a shared key $k_{x,y}$ with node y and distributes it to y, it uses y's public key (e.g., $K_{y,\mathrm{pub}}$) to encrypt $k_{x,y}$. By doing this, only y has a corresponding private key, e.g., $K_{y,\mathrm{pri}}$, to retrieve the key. Therefore, it is completely avoided for the key distribution to be compromised on the fly.

VI. Conclusion

This paper has proposed an attack detection and defense mechanism by using both the route redundancy in ad hoc networks and the message redundancy in topology discovery of the routing protocols. This paper also develops an optimal routing algorithm by combining both trustworthiness and performance. To our knowledge, this is the first secure routing that quantitatively considers not only the detection of difficult internal attacks but the network performance as well.

The simulation results have demonstrated the effectiveness of the proposed attack detection algorithm and optimal routing protocol, and superiority over such known protocols as AODV and Ariadne. The proposed attack detection and routing algorithms can be integrated into existing routing protocols for MANETs, such as AODV and DSR.

In this and other secure routing protocols, the computational burden at each node is still a major issue in deployment. It requires both analytical investigations and engineering considerations. For example, how many neighbors should a node have without degrading network performance and security? How many copies should a node receive before sending back an acknowledgement? Currently, SRAC considers the link performance as a routing metric. Considering the mobility in SRAC is expected to increase the prediction accuracy and thus reduce the link breakage rate during deployment. All these problems will further be investigated in future work.

Acknowledgment

The authors would like to thank the editors and reviewers, for their constructive comments and suggestions, and S. Kulkarni and T. Zhu, for conducting the simulations.

References

[1] P. Papadimitratos and Z. Haas, "Securing the Internet routing infrastructure," *IEEE Commun. Mag.*, vol. 40, no. 10, pp. 60–68, Oct. 2002.

[2] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Ad Hoc Netw.*, vol. 1, no. 2/3, pp. 293–315, Sep. 2003.

[3] C. Zhang, M. C. Zhou, and M. Yu, "Ad hoc network routing and security: A review," *Int. J. Commun. Syst.*, vol. 20, no. 8, pp. 909–925, Aug. 2007.

[4] J.-S. Hwu, R.-J. Chen, and Y.-B. Lin, "An efficient identity-based cryptosystem for end-to-end mobile security," *IEEE Trans. Wireless Commun.*, vol. 5, no. 9, pp. 2586–2593, Sep. 2006.

[5] C. E. Perkins and E. M. Royer, "The ad hoc on-demand distance-vector protocol," in *Ad Hoc Networking*, C. E. Perkins, Ed. Reading, MA: Addison-Wesley, 2001, ch. 6, pp. 173–220.

[6] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *Proc. ACM SIGCOMM Conf. Comms. Architectures, Protocols Appl.*, 1994, pp. 234–244.

[7] D. B. Johnson, D. A. Maltz, and J. Broch, "DSR: The dynamic source routing protocol for multihop wireless ad hoc networks," in *Ad Hoc Networking*, C. E. Perkins, Ed. Reading, MA: Addison-Wesley, 2001, ch. 5, pp. 139–172.

[8] M. G. Zapata and N. Asokan, "Securing ad hoc routing protocols," in *Proc. ACM WiSe*, Atlanta, GA, Sep. 28, 2002, pp. 1–10.

[9] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," in *Proc. 8th ACM Int. Conf. Mobile Comput. Netw.*, Sep. 2002, pp. 12–23.

[10] Y.-C. Hu, D. B. Johnson, and A. Perrig, "SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks," in *Proc. 4th IEEE Workshop Mobile Comput. Syst. Appl.*, Jun. 2002, pp. 3–13.

[11] K. Sanzgiri, D. LaFlamme, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer, "Authenticated routing for ad hoc networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 3, pp. 598–610, Mar. 2005.
[12] W. Liu, W. Lou, and Y. Fang, "An efficient quality of service routing algorithm for delay-sensitive applications," *Comput. Netw.*, vol. 47, no. 1, pp. 87–104, Jan. 2005.
[13] K. K. Leung and L.-C. Wang, "Integrated link adaptation and power control to improve error and throughput performance in broadband wireless packet networks," *IEEE Trans. Wireless Commun.*, vol. 1, no. 4, pp. 619–629, Oct. 2002.
[14] P. Papadimitratos and Z. Haas, "Secure routing for mobile ad hoc networks," in *Proc. SCS Commun. Netw. Distrib. Syst. Model. Simul. Conf.*, Jan. 2002, pp. 27–31.
[15] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "Efficient and secure source authentication for multicast," in *Proc. NDSS*, San Diego, CA, Feb. 8–9, 2001, pp. 90–100.
[16] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Rushing attacks and defense in wireless ad hoc network routing protocols," in *Proc. ACM WiSe*, Sep. 2003, pp. 30–40.
[17] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Packet leashes: A defense against wormhole attacks in wireless networks," in *Proc. INFOCOM*, Hong Kong, Apr. 2003, pp. 1976–1986.
[18] R. Perlman, "Network layer protocols with Byzantine robustness," Mass. Inst. Technol., Cambridge, MA, MIT LCS, TR-429, Oct. 1988.
[19] R. Perlman, "Routing with Byzantine robustness," Sun Microsyst., Santa Clara, CA, TR-2005-146, Sep. 2005. [Online]. Available: http://research.sun.com/techrep/2005/smli_tr-2005-146.pdf
[20] B. Awerbuch, R. Curtmola, D. Holmer, and C. Nita-Rotaru, *ODSBR: An On-Demand Secure Byzantine Routing Protocol*, Oct. 15, 2003, JHU CS Tech. Rep., Ver. 1. [Online]. Available: http://www.cnds.jhu.edu/research/networks/archipelago/publications/ODSBR-Awerbuch-TechReport1-2003.pdf
[21] I. Avramopoulos, H. Kobayashi, R. Wang, and A. Krishnamurthy, "Highly secure and efficient routing," in *Proc. IEEE INFOCOM*, Hong Kong, Mar. 2004, pp. 197–208.
[22] M. T. Goodrich, "Leap-frog packet linking and diverse key distributions for improved integrity in network broadcasts," in *Proc. IEEE Symp. Security Privacy*, 2005, pp. 196–207.
[23] D. Boneh, C. Gentry, H. Shacham, and B. Lynn, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proc. Eurocrypt*, 2003, vol. 2656, p. 641.
[24] S. Capkun, L. Buttyan, and J. Hubaux, "Self-organized public-key management for mobile ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 2, no. 1, pp. 1–13, Jan.–Mar. 2003.
[25] J. Dowling, E. Curran, R. Cunningham, and V. Cahil, "Using feedback in collaborative reinforcement learning to adaptively optimize MANET routing," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 35, no. 3, pp. 360–372, May 2005.
[26] B. Krishnamachari and S. Iyengar, "Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks," *IEEE Trans. Comput.*, vol. 53, no. 3, pp. 241–250, Mar. 2004.
[27] A. A. Pirzada and C. McDonald, "Establishing trust in pure ad-hoc networks," in *Proc. 27th ACSC*, 2004, pp. 47–54.
[28] T.-W. Kwon, C.-S. You, W.-S. Heo, Y.-K. Kang, and J.-R. Choi, "Two implementation methods of a 1024-bit RSA cryptoprocessor based on modified Montgomery algorithm," in *Proc. IEEE ISCAS*, May 6–9, 2001, vol. 4, pp. 650–653.
[29] Z. Wang, L. Liu, and M. C. Zhou, "Space and network diversity combination for masked node collision resolution in wireless ad hoc network," *IEEE Trans. Wireless Commun.*, vol. 6, no. 2, pp. 478–485, Feb. 2007.
[30] M. Yu, S. Kulkarni, and P. Lau, "A new secure routing protocol to defend Byzantine attacks for ad hoc networks," in *Proc. IEEE ICON*, Kuala Lumpur, Malaysia, Nov. 16–18, 2005, vol. 2, pp. 1126–1131.
[31] L. Peterson and B. Davie, *Computer Networks: A Systems Approach*, 3rd ed. San Mateo, CA: Morgan Kaufmann, May 2003, ch. 8.
[32] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Englewood Cliffs, NJ: Prentice–Hall, 1992, ch. 5.
[33] [Online]. Available: http://www.isi.edu/nsnam/ns/

Ming Yu (SM'05) received the Doctor of Engineering degree from Tsinghua University, Beijing, China, in 1994 and the Ph.D. degree from Rutgers University, New Brunswick, NJ, in 2002, both in electrical and computer engineering.

Since July 1997, he has been a Senior Technical Staff Member with AT&T, Middletown, NJ. From 1999, he was with the Department of ATM Network Services, AT&T Labs, Middletown. From December 2002, he was with the Department of IP/Data Network Management System Engineering, AT&T Labs. During September 2003 and August 2006, he was with the Department of Electrical and Computer Engineering, State University of New York, Binghamton. Since September 2006, he has been an Assistant Professor with the Department of Electrical and Computer Engineering, Florida State University, Tallahassee. His research interests are routing and MAC protocols, security, radio resource management, traffic modeling, and performance analysis for both wired and wireless networks.

Dr. Yu was awarded an IEEE Millennium Medal in May 2000.

Mengchu Zhou (S'88–M'90–SM'93–F'03) received the B.S. degree from Nanjing University of Science and Technology, Nanjing, China, the M.S. degree from Beijing Institute of Technology, Beijing, China, and the Ph.D. degree from Rensselaer Polytechnic Institute, Troy, NY.

Since 1990, he has been with the New Jersey Institute of Technology (NJIT), Newark, where he is currently a Professor of electrical and computer engineering and the Director of the Discrete-Event Systems Laboratory. His interests are in computer-integrated systems, Petri nets, networks, and automation.

Dr. Zhou is the Managing Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, and an Associate Editor of the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS. He was the recipient of the CIM University–LEAD Award from the Society of Manufacturing Engineers, the Perlis Research Award from NJIT, the Humboldt Research Award for US Senior Scientists, and the Distinguished Lecturer Award from IEEE SMC Society.

Wei Su (S'90–M'91–SM'95) received the B.S. degree in electrical engineering and the M.S. degree in systems engineering from Shanghai Jiao Tong University, Shanghai, China, in 1983 and 1987, respectively, and the Ph.D. degree in electrical engineering from The City University of New York, in 1992.

In 1994, he was an electronics engineer with the Army Research Laboratory, Fort Monmouth, NJ. Since 1998, he has been a Senior Technical Staff and Project Leader with the U.S. Army Communication Electronics Research Development and Engineering Center, Fort Monmouth. His research interests include communication intelligence, signal and image processing, and automatic control.

Dr. Su was the recipient of the 2002 Thomas Alva Edison Patent Award, the 2004 AOC International Research and Development Award, and the 2005 Army Research and Development Award.