# A Trustworthiness-Based QoS Routing Protocol for Wireless Ad Hoc Networks

Ming Yu, *Senior Member, IEEE,* and Kin K. Leung, *Fellow, IEEE*

***Abstract*—Due to the fact that the wireless links in an ad hoc network are susceptible to attacks and the nodal mobility renders the network to have a highly dynamic topology, it becomes critical to detect major attacks against the routing protocols of such networks and also provide some extent of QoS to the network traffic. In this paper, we present a new secure routing protocol (SRP) with quality of service (QoS) support, called Trustworthiness-based Quality Of Service (TQOS) routing, which includes secure route discovery, secure route setup, and trustworthiness-based QoS routing metrics. The routing control messages are secured by using both public and shared keys, which can be generated on-demand and maintained dynamically. The message exchanging mechanism also provides a way to detect attacks against routing protocols, particularly the most difficult internal attacks. The routing metrics are obtained by combing the requirements on the trustworthiness of the nodes in the network and the QoS of the links along a route. The simulation results have demonstrated the effectiveness of the proposed secure QoS routing protocol in both security and performance.**

***Index Terms*—Security, QoS, routing protocol, mobile ad hoc network.**

## I. Introduction

WIRELESS ad hoc networks start to be widely deployed in various environments. A particular challenging problem in designing such networks is how to detect the major attacks against the routing protocols while also provide some QoS support to the network traffic.

First, the routing protocols must be secured to defend attacks that may come from external or internal nodes. In an external attack, a malicious node masquerades as a trusted node although it does not participate in the routing process. It can generate floods of spurious service requests, such as denial of service (DOS) attack. While in an internal attack, a malicious node may be a compromised or misconfigured node participating routing, or even colludes with other malicious nodes, which is called a Byzantine attack [4]. It may advertise false routing information, not forward packet correctly, misroute, fabricate, modify, or simply drop packets [22]. Therefore, it is more difficult to detect the internal attacks. Second, the protocols must be integrated with QoS routing schemes to support the QoS requirements of the carried traffic, for example, to minimize a cost under delay constraint [2], or minmax a cost caused by a single link failure [3] or by co-channel interferences [24].

The existing SRPs for ad hoc networks often avoid either the most challenging internal attacks, such as Byzantine behaviors, or the QoS requirements of the traffic.

### A. Related Work

The existing SRPs for ad hoc networks can be divided into two categories: in terms of how an SRP is secured and what types of attacks it can defend.

In the first category, the commonly used method is to establish a security association between the source and destination nodes so that the on-demand routing protocols, such as AODV, DSR, and DSDV, can be secured [5]. In [7], the authors proposed an SRP called Ariadne based on DSR by using efficient symmetric cryptography. Routing messages are authenticated by shared secrets between each pair of nodes. The broadcast authentication scheme used in Ariadne is TESLA [8], which requires loose time synchronization. In [9], the authors proposed a proactive SRP, called SEAD, based on DSDV by using one-way hash chains to provide authentication to defend attacks that modify routing information broadcast and replay attacks but not wormhole attack. In [10], in order to secure on-demand protocols such as AODV and DSR, the authors developed an authenticated routing protocol, called ARAN, by using digital signature to provide end-to-end authentication, message integrity, and nonrepudiation. During route discovery, a routing message is signed by a source node and then broadcasted to others. An intermediate node that receives the message will replace the certificate and signature of the previous hop with those of its own and then forwards the message to the next hop. During route setup, the message is similarly signed twice and unicasted back to the source. Due to its use of double signatures, ARAN can defend most common attacks.

In the second category, the major purpose is to protect routing traffic against the internal attacks, particularly Byzantine attacks [4]. In [23], the authors proposed to use both route and message redundancy to detect Byzantine behaviors by comparing different copies of a message received over different routes. In [11], the authors proposed to detect Byzantine behaviors by using a probing technique, which uses binary search on a path to find out faulty links. The accumulated path is protected by an aggregate signature scheme [12], which is even more expensive than RSA signatures. In [13], the authors proposed an SRP against Byzantine failures by using source routing and destination acknowledgements. Each packet is authenticated at each node by using MACs based on pair-wise

Manuscript received February 3, 2008; revised June 23, 2008; accepted August 6, 2008. The associate editor coordinating the review of this paper and approving it for publication was W. Zhuang.

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

K. K. Leung is with the Department of Electrical & Electronic Engineering, Imperial College, London, UK (e-mail: kin.leung@imperial.ac.uk).

Digital Object Identifier 10.1109/TWC.2009.080161

TABLE I
NOTATIONS FOR SECURITY PRIMITIVES

| Notation | Meaning |
|---|---|
| $K_{A+}$ | Public key of node $A$ |
| $K_{A-}$ | Private key of node $A$ |
| $K_{AB}$ | Symmetric key shared by nodes $A$ and $B$ |
| $\{d\}K_{A+}$ | Encryption of data $d$ with key $K_{A+}$ |
| $[d]K_{A-}$ | Data $d$ digitally signed by node $A$ |
| $cert_A$ | Certificate belonging to node $A$ |
| $N_A$ | Nonce issued by node $A$ |
| $IP_A$ | IP address of node $A$ |
| $P_A$ | Routing Packet sent out by node $A$ |
| $F_{I_j}$ | Flag to indicate node $I_j$'s misbehavior |
| RDP | Route Discovery Packet identifier |
| REP | REply Packet identifier |
| RER | Route ERror packet identifier |

secret keys. Digital signatures are used for initial key setup. Misbehaviors are detected on a per packet basis to defend Byzantine adversaries.

Based on the above review, we conclude that few protocols are capable of detecting internal attacks such as Byzantine attacks and use expensive aggregate signatures or per packet filtering. Also, the routing metrics are still performance indicators, not security metrics, such as the trustworthiness of a node. Therefore, few routing protocols consider the security and QoS problems together.

### B. Contribution of This Work

For these reasons, we propose a new secure QoS routing protocol, i.e., TQOS. First, it is able to detect the difficult internal attacks, including Byzantine attacks. Second, the results on the message verification conducted by a node are used to build a trustworthiness repository by the node on its neighboring nodes that deliver the message. Third, the trustworthiness is incorporated into the routing metrics, which contains the QoS requirement on the links along a route, such as packet delay and link quality.

The paper is organized as follows. The secure route discovery and key setup are presented in in Section II. The trustworthiness-based QoS routing metrics and algorithm are discussed in Section III. The security, cost, and impact of TQOS are analyzed in Section IV. The simulation results are presented in Section V. We conclude this paper in Section VI.

## II. SECURE MESSAGE EXCHANGE AND KEY DISTRIBUTIONS

We assumed that each node initially has a pair of public/private keys issued by a public key infrastructure (PKI) or other certificate authority (CA) during its deployment. Each node maintains a local certificate repository (CR), which can be managed by using the trustworthiness information and used to build up a self-organized PKI, as in [14], which in turn can be used to create and distribute the shared keys among the nodes [1], [15]. Unlike many other SRPs, our framework of attack detection does not rely on the PKI or CA after the deployment of the nodes.

For convenience, the notations for security primitives are summarized in Table I, which are the same as those used in ARAN so that comparisons can be easily made.

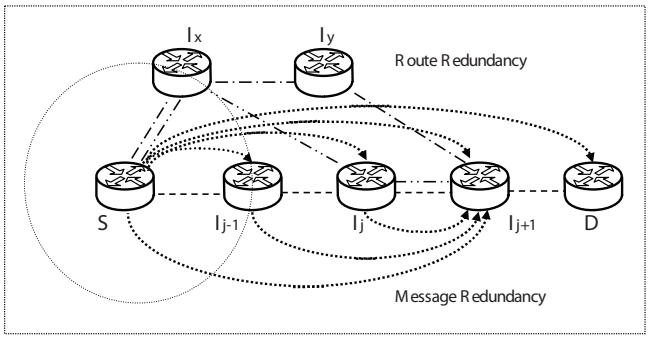


Fig. 1. Multiple copies of a message forwarded over different routes

Initially, each node in the network receives a certificate from a CA, denoted by $T$. For example, for node $A$, the certificate has the following format:

$$T \rightarrow A : cert_A = [IP_A, K_{A+}]K_{T-}, K_{T+}. \quad (1)$$

The certificate contains the IP address of $A$ (e.g., $IP_A$) and the public keys of $A$ (e.g., $K_{A+}$) and $T$ (e.g., $K_{T+}$). In the above notation, the messages inside $[\ldots]$, e.g., $IP_A$ and $K_{A+}$, are concatenated and signed by $T$. The signed message is then concatenated with $K_{T+}$, e.g., the public key of $T$, as the certificate to $A$. To simplify the notation, we ignore the timestamp related information.

### A. Secure Route Discovery

Before a source node $S$ can send data packets to a destination node $D$, it needs to discover the network topology by sending out a *route discovery packet* (RDP) in order to setup a route between $S$ and $D$. The discovery process is demonstrated in Fig. 1.

To simplify notations, we define a format for an RDP message sent out by $S$,

$$m_d = RDP, IP_D, N_S, \quad (2)$$

where the packet type identifier (e.g., RDP), destination's IP address (e.g., $IP_D$), and a nonce (e.g., $N_S$) are all concatenated. Note that $N_S$ works like a sequence number. For example, we choose a size of 5 bytes for $N_S$ in our simulations, same as in ARAN.

Similarly, we define a format for a route reply (REP) packet replied by $D$:

$$m_r = REP, IP_S, N_S, \quad (3)$$

where the packet type identifier (e.g., REP), source's IP address (e.g., $IP_S$), and a nonce (e.g., $N_S$) are also concatenated.

At first $S$ checks whether or not exists a shared key between $S$ and $D$, i.e., $K_{SD}$. If it does not exist, $S$ can create one and encrypt it with $D$'s public key:

$$m_D = m_d, \{m_d\}K_{SD}, \{K_{SD}\}K_{D+}. \quad (4)$$

Only $D$ that has its private key can decrypt the message to find the shared key and validate the message. Others can only view the message. Both $m_d$ and its encrypted version $\{m_d\}K_{SD}$ are sent out, so that intermediate nodes can view but cannot modify the message. Only the destination can verify the correctness of the message.

In the next section, we will see that the shared key can also be established by using the security associations between intermediate nodes in the route setup phase. If $S$ already has a shared key with $D$, the third part in $m_D$ defined in Eqn. (4) can be simply removed.

In order to discover its neighboring nodes, $S$ broadcasts the RDP packet as follows:

$$S \rightarrow broadcast : P_S = [m_D]K_{S-}, cert_S. \quad (5)$$

Since $S$'s certificate (e.g., $cert_S$) is also appended to the packet, any node that receives the packet also receives $S$'s public key, which has been certified by $T$.

When an intermediate node, say $I_j$, as shown in Fig. 1, receives the RDP from its predecessor, i.e., $I_{j-1}$, it validates the signature of the message. If it is valid, then $I_{j-1}$ is a valid neighbor of $I_j$. If not, $I_{j-1}$'s trustworthiness by $I_j$ will be updated. It can use Eqn. (15) to update $\hat{T}_{I_{j-1}}(I_j; x)$, which is $I_{j-1}$'s trustworthiness value observed by $I_j$ during $x$-th route discovery cycle. If $\hat{T}_{I_{j-1}}(I_j; x)$ is below a threshold, it thinks that $I_{j-1}$ is untrustworthy, and its repository of trustworthiness will be updated.

As a neighbor of $S$, $I_{j-1}$ receives a copy of the broadcasted message. It validates that the message comes from $S$ and then rebroadcasts it in the following format:

$$I_{j-1} \rightarrow broadcast : P_{I_{j-1}} = [m_D]K_{I_{j-1}-}, cert_{I_{j-1}}. \quad (6)$$

In the above format, $I_{j-1}$ signs the RDP packet and attaches its own certificate to the message.

Similarly, as one of $I_{j-1}$'s neighbors that are discovered in the previous step, $I_j$ also rebroadcasts the RDP in the following format:

$$I_j \rightarrow broadcast : P_{I_j} = [m_D]K_{I_j-}, cert_{I_j}. \quad (7)$$

In the rebroadcast, each $I_j$ remembers that $I_{j-1}$ is its predecessor. The rebroadcast continues for those have not been labeled as predecessors until all the nodes in the network are discovered.

If $I_j$ has shared keys with its known neighbors, for example, $K_{I_jI_{j+1}}$ with $I_{j+1}$, or $K_{I_jI'_{j+1}}$ with $I'_{j+1}$, where $I_{j+1}$ and $I'_{j+1}$ are the two neighbors of $I_j$, it sends out the RDP to the two nodes as follows:

$$I_j \rightarrow I_{j+1}/I'_{j+1} : P_{I_j} = m_D, \{m_D\}K_{I_jI_{j+1}}/\{m_D\}K_{I_jI'_{j+1}}. \quad (8)$$

It can be seen that a message $m_d$ is broadcasted by the source (see Eqn. (5)), and rebroadcasted by each intermediate nodes (see Eqns. (6) and (7)), or multicasted by an intermediate nodes (see Eqn. (8)). Thus, any node may receive multiple copies of the same message, which is called *message redundancy*. More over, the same message may arrive a node via multiple routes, which is called *route redundancy*. In this way, a destination will be able to not only find the discrepancies among different copies of a message, but also discover that which node does not follow the protocols, i.e., behaves abnormally. For example, a selective forwarding attack can be detected by a receiving node, after it compares the multiple copies of a message received from multiple nodes and finds out which one is more trustworthy [23].

### B. Secure Route Setup

The route setup procedures can be summarized as follows. When $D$ receives an RDP packet from a neighbor node, it first checks whether exists a shared key between $D$ and the node. If the answer is no, $D$ creates such a key and delivers it to the neighbor node. Then, it sends out an REP packet to the node, which forwards the packet until the packet reaches the source node. During the establishment of the route, the trustworthiness is continuously updated.

To build such a shared key with its neighbors, $D$ sends out an REP packet in the following format:

$$D \rightarrow I_{j+1} : P_D = [m_R, \{K_{I_{j+1}D}\}K_{I_{j+1}+}]K_{D-}, cert_D. \quad (9)$$

where

$$m_R = m_r, \{m_r\}K_{SD}, \{K_{SD}\}K_{S+}.$$

In the above format, a shared key, e.g., $K_{I_{j+1}D}$, is created by $D$ using a pseudorandom key generator, and then encrypted by using $I_{j+1}$'s public key. Similarly, the source's public key is used to encrypt $K_{SD}$, so that only the source can decrypt it by using its private key. The whole message is then signed by using the destination's private key. Any node that receives the message can be assured that the message is from $D$ and can only read the message but cannot modify it because it is protected by the shared key, which is encrypted.

The purpose for sending out the above packet is that the destination can build the shared keys between itself and the intermediate nodes, through which the shared key $K_{SD}$ is delivered to $S$ hop by hop. Note that $K_{SD}$ will be used as a session key during data transmissions once the route is established.

When an intermediate node $I_{j+1}$ receives the REP, it finds the shared key, i.e., $K_{I_{j+1}D}$, and stores it for data transmissions between $I_{j+1}$ and $D$. It then forwards the REP to the next hop as follows:

$$I_{j+1} \rightarrow I_j : P_{I_{j+1}} = [m_R, F_{I_{j+1}}, \{K_{I_jI_{j+1}}\}K_{I_j+}]K_{I_j-}, cert_{I_j}. \quad (10)$$

where the REP from the node $I_{j+1}$ is forwarded to the next node $I_j$. Also included is a flag, e.g., $F_{I_{j+1}}$, to indicate whether or not an intermediate node is detected with a misbehavior. In our simulations, $F_{I_{j+1}}$ is simply implemented as an error code. Once the security association between the two nodes has been established, $I_j$ can update its trustworthiness and choose a route of minimum cost.

In this way, hop by hop, the REP reaches the source $S$ that sent out the original RDP and thus a security association between $S$ and $D$ is established by sharing a symmetric key $K_{SD}$. At the same time, an intermediate node also establishes a security association with its neighbors.

Once $K_{SD}$ is established, $D$ sends out the REP in the following format during another round of route setup:

$$D \rightarrow I_{j+1} : P_D = m_R, \{m_R\}K_{I_{j+1}D}, \quad (11)$$

where

$$m_R = m_r, \{m_r\}K_{SD}, \{K_{I_jD}\}K_{I_j+},$$

where the message is protected by the shared key $K_{SD}$. At the same time, another shared key is created for $I_j$ and $D$

and encrypted by $I_j$'s public key. Similarly, $I_{j+1}$ sends out the REP in the following format:

$$I_{j+1} \rightarrow I_j : P_{I_{j+1}} = m_R, F_{I_{j+1}}, \{m_R, F_{I_{j+1}}\}K_{I_j I_{j+1}}, \quad (12)$$

where the misbehavior flag $F_{I_{j+1}}$ is appended to the message, if $F_{I_{j+1}}$ exists. When $I_j$ receives the REP, it finds the shared key with $D$. In the same way, $I_{j-1}$ can also establish a shared key with $D$, and so on. Here, the redundant use of shared keys is for attack detection purpose.

For example, $I_{j+1}$ can establish three shared keys with $I_j$, $I_{j-1}$, and $S$, respectively, as shown in Fig. 1. For a same message sent out by $S$, $I_{j+1}$ can receive three copies that are encrypted with the three keys. By comparing the discrepancies among the three copies, $I_{j+1}$ can find out which one is more trustworthy and the rest are compromised.

Once the security associations have been built between the source and destination, and a trustworthy route has been established, a source can simply use an efficient shared key to send data, for example, $m_{data}$:

$$P_{data} = m_{data}, \{m_{data}\}K_{SD}. \quad (13)$$

In this work, both shared keys and public/private keys are used in protecting messages. The shared key is used as a session key to verify message integrity, particularly the data, while the public/private keys are used to authenticate nodes and establish the session key. The reason is that the session keys can be updated more frequently than the public/private keys. It is known that the symmetric algorithm for the shared key, e.g., keyed-hashed MAC, such as MD5, is much more efficient than the asymmetric algorithm for the public/private keys, such as RSA. For example, MD5 can process data at a typical speed of 600 Mbps, which is about 1000 times faster than RSA [26].

It is worth pointing out that the identity-based encryption (IBE) can be an efficient alternative to PKI, because a public key can be generated on-demand by cryptographically combining a node's identity [27]. For ad hoc networks, the relying of IBE on a centralized and trusted private key generator needs to be removed, which is similar to build up a self-organized PKI used in this work.

## III. Trustworthiness-based QoS Routing Protocol

We assume that each node has built up a trust repository locally for the nodes it knows based on its certificate repository and current behavior that has been observed during topology discovery. The repository can be updated every time the topology is rediscovered, as discussed in Section II.

### A. Routing Metrics and Optimal Routing

Let us assume that node $j$ is one of node $i$'s neighbors. Denote by $\hat{T}_j(i;n)$ the trust on node $j$, assigned by node $i$, after the $n$-th topology updating cycle. Note that a simple way to measure $\hat{T}_j(i;n)$ is to compute the ratio of the number of messages decoded and verified correctly to the number of messages that have been received. For example, one such method is to use a statistics model as in [17]:

$$T_j(i;n) = N_{VER}(i;n)/N_{REC}(i;n),\ \mathrm{j} \in \{i\text{'s neighbors}\}, \quad (14)$$

where $N_{VER}$ and $N_{REC}$ are the numbers of messages that have been verified by $i$ and received from $j$ at time $n$, respectively. Note that $N_{VER}$ is counted based on the secure message verification, indicated by $F_{I_j}$, as discussed in Section II.

Every time a new observation comes in, the node updates its repository and calculate a trust value by using a moving average model. After the $(n+1)$-th topology updating cycle, we can get

$$\hat{T}_j(i;n+1) = \gamma \hat{T}_j(i;n) + (1-\gamma)\tilde{T}_j(i;n+1), \quad (15)$$

where $\tilde{T}_j(i;n+1)$ is node $j$'s trust value measured by node $i$ during the $(n+1)$-th topology updating cycle; $0 < \gamma < 1$ is a weighting factor used to trade off between current measurement and previous estimation.

Consider a path $p$ that starts from a source node $s$ to a destination node $d$, i.e., $p \in \Omega_{sd}$={all the paths from $s$ to $d$}. Let's denote by $T_p(n)$ the trust value of the path $p$ at time $n$. In path $p$, each node chooses one node from its routing table as its next hop to forward an REP packet. The source $s$ uses this path to send data packets. $T_p(n)$ can be expressed as

$$T_p(n) = \prod_{i,j \in p} \hat{T}_j(i;n), \quad (16)$$

which serves as a security requirement on path $p$.

To consider link quality, we use the expected transmission count (ETX) as a metric, which was proposed in [18]. It estimates the number of retransmissions required to send packets by measuring the loss rate of broadcast packets between pairs of neighboring nodes. In order to calculate the ETX metric, each node broadcasts a probe packet in a fixed time interval. Each node also sends out a packet containing the count of probe packets received from each of its neighboring node in the previous time interval. Based on these probes packets, a node calculates the loss rate of probe packets on the links to and from its neighbors. Since our routing protocol uses hello packets for initial neighbor discovery, we can modify the hello packets to include some extra fields, such as the number of hello packets received in last interval. In this way, each node obtains a value for the ETX based on its observations.

Let's denote by $X_j(n)$ the ETX measured by node $j$ at time $n$, which can be expressed as

$$X_j(n) = \frac{1}{FDR_j(n) \times RDR_j(n)}, \quad (17)$$

where $FDR_j(n)$ is the forward delivery ratio at time $n$, i.e., the ratio of the number of packets received by node $j$ to the total packets it sends out; $RDR_j(n)$ is the reverse delivery ratio at time $n$, i.e., the probability that an ACK packet is successfully received by $j$. Thus, $FDR_j(n) \times RDR_j(n)$ is the chance that a probe packet is successfully sent to and acknowledged back by a receiving node. Therefore, $X_j(n)$ represents the expected number of transmissions for node $j$ to successfully deliver a packet.

To consider QoS requirements, for example, packet delay for delay-sensitive traffic, we use $\tau_j$ to represent the delay a packet experienced when being delivered over a link starting from node $j$. Thus, we can design a combined link cost as follows:

$$\Psi_j(n) = \alpha_j X_j(n) \tau_j, \quad \text{for } j \in p, \quad (18)$$

where $0 < \alpha_n < 1$ is a scaling factor used to scale the impact of the amount of delay on the total metric. For simplicity, we can choose $\alpha_j = \alpha$ for all $j$. Clearly, $\Psi_j(n)$ represents an expected total delay of a packet being delivered over the link a few times until an ACK being successfully received by node $j$.

Similarly, we can design a combined metric for path $p$ as follows:

$$\Psi_p(n) = \sum_{j \in p} \alpha_j X_j(n) \tau_j, \quad \text{for } p \in \Omega_{sd}. \tag{19}$$

It can be seen that $\Psi_p(n)$ represents a weighting average of the total delay of a packet being delivered from a source to its destination node along a specific path $p$. Therefore, it can be used as a cost that penalizes the paths with relatively long delays among those between the source and destination.

Considering both the trustworthiness and QoS requirements, a path that is less trusted and does not meet the desired QoS must be penalized in our objective, thus a combined cost function can be designed as

$$C_p(n) = \sum_{j \in p} \beta \Psi_j(n)(1 - T_j(n)), \quad \text{for } p \in \Omega_{sd}, \tag{20}$$

where $0 < \beta < 1$ is a coefficient used to scale the cost during optimization although it has no impact on the optimization results. It can be seen that a bigger value of $C_p(n)$ means a higher cost for path $p$ to be selected, which is due to either the lower quality of the path, or the less trustworthiness of the path, or both.

Now the routing problem can be written as:

$$\min_p \ C_p(n), \quad \text{for } p \in \Omega_{sd}, \tag{21}$$

and subject to the constraints:

$$\hat{T}_j(i;n) \geq T_{min}; \quad \text{for } i, j \in p, \tag{22}$$

where $T_{min}$ is the minimum trust required for a node to be allowed to join a route. In Eqn. (20), we explicitly integrate the security and QoS requirements. For each node $j$ on path $p$, the values of $X_j$, $\tau_j$ and $\hat{T}_j(i;n)$ can be obtained. Therefore, node $j$ can calculate an increase to the cost function $C_p(n)$ if its next hops is selected to join the path in order to have a minimum $C(p)$.

To find a global optimal solution to the above optimal routing problem by solving the Eqns. (21) and (22), there exist many classical algorithms, including Dijkstra algorithm and distributed Bellman-Ford algorithm [19]. But using these algorithms, a source node needs to conduct extensive communications with other nodes, which can be significantly expansive due to the use of secure message exchanges.

### B. Routing Algorithm

The heuristic algorithm can be summarized as follows.

1. During route discovery, a source node broadcasts the RDP packets. In these packets, along with the regular routing information, their keys are also attached in order to establish a security association with their neighbors. Also included is their observation on the trustworthiness of their neighboring nodes, and the related misbehavior information.

2. Once an intermediate node receives the RDP packet, it verifies the source, validates the packet, acknowledges its previous hop, and establishes a pair-wise shared key with the previous hops and the source. It then replaces some fields in the RDP packets by its encryption of keys and adds to the RDP packet and forwards it to its next hops. This process is repeated until it either reaches the destination or an intermediate node that has established a route to the destination.

3. At the destination, it verifies the packets and retrieves key information to establish a pair-wise shared key with the intermediate nodes and the source. The total cost for the route will be calculated by the destination by using Eqn. (20). Note that the destination will not make a routing decision until it receives a few valid copies of a same message over multiple routes. Once a enough number of RDP packets reach the destination, the destination compares the multiple routes and selects the one with the least cost. The destination node then sends an REP back to the source node over the selected route.

4. When the source node receives the REP, it verifies the packet and retrieves the key information, particularly the shared key with the destination. Once the security association is established, it starts data transmissions on the route. The intermediate nodes on the route monitor the link status and observes the behavior of their neighbors. It reports the misbehavior by appending a misbehavior indicator to the packet.

5. When a link status error or security violation is detected, a route error (RER) message is generated to notify other nodes so that they can update their trustworthiness on the erroneous node or link. The RER message indicates that the erroneous node or link must be avoided if the trustworthiness is below a certain threshold, as stated in Eqn. (22). Meanwhile, a new route which bypasses the erroneous node is selected from the already existing set of candidate routes. If no such route exists then a new route discovery is conducted.

Comparing to existing secure routing algorithms, the proposed algorithm uses both digital signature and encryption, in stead of using double signatures, to protect packets from internal attacks. It also explicitly incorporates trustworthiness, link quality, and QoS requirements into the routing metrics.

## IV. Security and Cost Analysis

In this section, we analyze the security of TQOS in the presence of different attacks. We compare it to ARAN in defending the difficult internal attacks in a network. We also evaluate the computational and energy cost incurred by the secure routing mechanism.

### A. Security Analysis

It is known that the majority of external attacks against routing protocols can be prevented by simple link layer encryption and authentication. In this paper, a source node shares multiple symmetric keys with a destination node and the intermediate nodes on a route between them. During the route discovery phase, every node authenticates its neighbors by using the RSA encryption algorithm. Hence, any malicious node that is not authenticated during the initial authentication phase is excluded from the subsequent route discovery and data transmissions.

*Sybil Attacks:* By using the shared keys to protect the routing information, Sybil attacks can be prevented because the nodes in the network will not accept any ID of the adversary nodes [16].

*Selective Forwarding and Sinkhole Attacks*: The majority of these attacks can be prevented because an adversary node is not allowed to join the network during topology discovery.

*HELLO Flood Attacks*: By using the shared keys between a node and its neighbors, HELLO flood attacks are prevented because the node can authenticate each of its neighbors.

The major classes of attacks not countered by the link layer mechanism are wormhole attacks and internal attacks, including Byzantine attacks. For this reason, TQOS uses multiple keys and digital signatures to protect the routing information and shared keys.

During the operations of TQOS, each node monitors its neighbors by comparing a message it receives over various paths. Each node maintains a repository of trustworthiness on its neighbors. If some authorized nodes do not forward the message correctly, i.e., become compromised, a receiving node can determine which of the nodes is compromised by comparing the multiple copies of the messages it has received over multiple paths, provided that there are enough number of uncompromised nodes. Otherwise, the network will not be functional. Thereafter, the compromised nodes can be excluded from the operations. In this way, the majority of *internal attacks* against routing protocols can be detected and prevented.

As an example, we discuss the detection of Byzantine attacks, which is particularly challenging among the internal attacks.

*Byzantine Attacks*: It is assumed that a receiving node receives multiple copies of a message, which are forwarded by different nodes over different routes that are protected by different keys. By comparing the copies of the message, the node finds discrepancies among the copies. It then chooses a correct one or more trusted one by using methods such as voting algorithms or Bayesian estimation [25], or simply the trustworthiness of the nodes who forward the copies of the message. For example, if some of the nodes that forward the message are more trustworthy than others, it can be claimed that those that have lower trustworthiness and forward the message differently are colluded nodes [23]. Here, the error in a copy of the message may be caused by a link error or a protocol violation, but the result is the same: the forwarded messages are not consistent.

In ARAN, a path is chosen by the RDP that reaches the destination first. The destination then sends back an REP along the reverse path to the source. The routing decision is based on the receiving of a single copy of the RDP or the sending of a single copy of the REP. If there are two or more nodes on a path that are compromised and colluding, the protocol may not be able to detect the problem. For example, as shown in Fig. 1, node $I_j$ is the next hop of $I_{j-1}$ to the destination $D$. If $I_{j-1}$ does not follow the protocol, for example, it does not sign the RDP packet and just forwards it to $I_{j+1}$, $I_j$ will detect the problem. But if $I_j$ is colluding with $I_{j-1}$, then the problem will not be found. The result is that the route that goes through the two nodes will be preferred. Once the route has been chosen by the destination, one of the two nodes can start to drop the REP packet. The problem will not be detected by the protocol. In TQOS, $I_{j+1}$ may receive three copies of the same RDP and thus can choose the one that is more trustworthy. The collusion will be detected once $I_{j+1}$ finds out the discrepancies among the copies of the message.

Compared to ARAN, TQOS has approximately the same computational cost at each node. But TQOS can defend some difficult internal attacks by using message and route redundancy. Compared to other existing secure routing protocols (e. g., [7], [9]), TQOS has a higher computational cost at each node in providing a higher level of security.

### B. Cost Analysis

In this section, we analysis the additional cost in processing the routing packets due to the use of the security mechanisms.

As we see in the previous sections, there are three basic security operations: signing and verifying a routing (control) packet with a private key, e.g., Eqn. (5); encrypting and decrypting a message with a shared key, e.g., Eqn. (8); and encrypting and decrypting a shared key with a public key, e.g., Eqn.(4). The routing (or control) packets, including RDP, REP, and RER are protected by a combination of the three operations, while data packets are only protected with the shared key.

We assume that the shared key algorithm uses a keyed-hashed MAC such as MD5 while the public/private algorithm uses RSA. Let's first estimate the extra time needed by using a single operation. Let's denote by $t_h$ the time for one hash operation, $t_g$ the time for one signature generation, $t_v$ the time for one signature verification, $t_e$ the time for one public key encryption, and $t_d$ the time for one public key decryption.

For the shared key operations, such as MD5, packets can be processed at a typical speed of 600 Mbps. For a typical message of 1 Kb, it takes $t_h = 1.6276\ \mu s$ to perform a hash operation. Thus, the time can be ignored.

For the public/private key operations, such as RSA, it is estimated that for a key with a length of $L_k$ (bits), the CPU cycles to perform one RSA operation is about $(L_k+2)\times(L_k+2+32)$ for a typical implementation [21], which equals $0.28$ and $1.09$ millions for $L_k = 512$ and $1024$, respectively. It is also estimated that the generation of a signature takes about $20$ RSA operations while the verification takes only one RSA operation. For a CPU of 2.8 GHz (e.g., an Intel Pentium 4 processor), we can find $t_g = 2.00$ and $7.80\ ms$, $t_v = 0.10$ and $0.39\ ms$, 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 $t_g = 27.00$ and $105.44\ ms$, $t_v = 1.35$ and $5.27\ ms$, for $L_k = 512$ and $1024$, respectively.

We also assume that the times spent in public key encryption/decryption are the same as those for the signature generation and verification. Thus, we have $t_e = t_g$ and $t_d = t_v$.

To process a routing packet (e.g., RDP), a node usually conducts a hash operation on the message, signs the message, and verifies that the certificate attached by a neighbor is valid, in order to replace the certificate with its own certificate in the attachment, as can be seen in Eqn. (5). Let's denote by $T_1$ the

TABLE II
SIMULATION PARAMETERS AND VALUES

| Parameters | Values |
|---|---|
| network dimensions | 670mx670m or 1000mx1000m |
| number of nodes | 20 or 50 |
| radio range | 250m |
| node pause times | 30s |
| node speed | $0 \sim 10\ m/s$ |
| source-destination pairs | 5 or 20 |
| traffic pattern | CBR/UDP |
| source traffic (each) | 4 packets/second |
| data payload size | 512 bytes/packet |

total time spent by a node for the security operations in this case. Therefore, $T_1 = t_g + t_v + t_h$. For a CPU of 2.8 GHz, we have $T_1 = 2.10$ and $8.19\ ms$, for $L_k = 512$ and 1024, respectively. For a CPU of 206 MHz, we have $T_1 = 28.35$ and $110.71\ ms$, for $L_k = 512$ and 1024, respectively.

To processing a routing packet (e.g., REP), if a shared key needs to be protected, a node needs to conduct one public key operation, e.g., Eqns. (11) and (12), or one public key and one signature operation, e.g., Eqn. (5), or even two public key encryptions plus one signature generation and verification, e.g., Eqn. (9). Therefore, the time to process an REP packet can be roughly $1 \sim 3$ times of that for an RDP packet. Let's denote by $T_2$ the total time spent by a node for the security operations in this case. For a CPU of 2.8 GHz, we have $T_2 = 2.10 \sim 6.30$ and $8.19 \sim 24.57\ ms$, for $L_k = 512$ and 1024, respectively. For a CPU of 206 MHz, we have $T_2 = 28.35 \sim 85.05$ and $110.71 \sim 332.13\ ms$, for $L_k = 512$ and 1024, respectively.

To processing a routing packet (e.g., RER), once all the pairwise shared keys are available, a node needs only to conduct a hash operation. Thus, the time is ignored.

### C. Impact on Network Performance

Now we can estimate the major impact of added security on network performance.

The first one is the part of the routing acquisition delay caused by the extra security operations, which is denoted by $T_{so}$. We assume that a discovered route has $h$ hops between a source and destination node. The destination needs to receive $c$ copies of the RDP packets until it makes a routing decision by sending an REP packet. Therefore, $T_{so} = c(h+1)T_1 + (h+1)T_2 = (h+1)(cT_1 + T_2)$. For the network configuration shown in Table II, by simulation we find that $c = 2 \sim 4$ and $h = 1 \sim 4$ hops. For a CPU of 2.8 GHz, $L_k = 512$, we have $T_{so} = 12.60 \sim 73.50\ ms$, which is acceptable for a typical setting of initial route request timeout = 2 $s$ and maximum route request timeout = 40 $s$. However, for a CPU of 206 MHz, $L_k = 1024$, we have $T_{so} = 0.6643 \sim 3.8749\ s$, which will significantly degrade the network performance and thus may restrict the applications of the secure routing protocol. Otherwise, we need to trade off the security requirement and network performance by reducing the length of the RSA key. For example, for a CPU of 206 MHz, if we choose $L_k = 256$, we can find that $T_1 = 15.25\ ms$ and $T_{so} = 91.50 \sim 533.75\ ms$, which will significantly improve the network performance.

The second one is on the network lifetime that is related to the energy consumption during route discovery and setup phases. This is critical because the nodes in a wireless ad hoc network are usually operated on battery power. Our concern is the extra power consumption due to the introduction of the security operations. It can be divided into two parts: the power consumed by the CPU and the wireless connection (e.g., NIC, network interface card). Let's denote by $E_{new}$ the energy in a new battery in mWh or mWs, $P_{CPU}$ the power dissipation by a CPU in mW, $P_{NIC}$ the power dissipation by a NIC in mW, $P_{tot}$ the total power dissipation in mW, $f_{CPU}$ the fraction of the power dissipation by a CPU in percentage, and $f_{NIC}$ the fraction of the power dissipation by a NIC in percentage. Usually $P_{NIC}$ is much larger than $P_{CPU}$. For example, for an iPAQ with a CPU of 206 MHz, powered by a 2000 mAh battery with a nominal voltage of 3.7 V, we have $E_{new} = 7400$ mAh. It is found that $f_{CPU} = 12\%$ and $P_{tot} = 1250$ mW without wireless connection while $f_{NIC} = 40\%$ and $P_{tot} = 2200$ mW with the connection [28]. For the two $P_{tot}$ values, the run times are 5.92 and 3.36 hours, respectively.

Due to the security mechanisms, the extra time for the CPU to process the routing packets is $T_{so}$. Every time the network topology is changed, the routes need to be updated once. Let's denote by $T_{ru}$ the interval of the route updating in s. Within an hour, the original power consumption is $P_{tot} \times 3600$, while the increased consumption is $T_{so} \times f_{CPU} \times P_{tot} \times 3600/T_{ru}$. Therefore, the increased consumption in an hour is $T_{so} \times f_{CPU}/T_{ru}$ in percentage. For example, for the iPAQ, if we choose $L_k = 256$ and $T_{ru} = 300$ s, we can find the increased consumption is $0.00366 \sim 0.02135\%$, which is quite acceptable.

It is noticed that the NIC card must be in awake mode at all times. To be able to receive and transmit packets, the NIC must be in receive or transmit state, in stead of idle or sleep. To find the increased power consumption by the NIC due to the use of security mechanisms, we need to find the increased operation time in receive or transmit state, which is also $T_{so}$. Similarly, the increased power consumption in an hour is $T_{so} \times f_{NIC}/T_{ru}$ in percentage. For the above setting, it is $0.0122 \sim 0.07117\%$, which is also acceptable. Note that the message redundancy introduced in route setup does not cost extra communications. The RDP packets are broadcasted to the whole network in both cases of without and with the security mechanisms. With the security mechanisms, a destination needs to delay the sending of an REP packet until receives a few copies of the RDP packets, in order to use the message redundancy. As we can see, the impact of the message redundancy is mainly on the additional routing delay, for which we need to trade off the key length and the routing delay. In our simulations, the number of copies is found to be in the range of $2 \sim 4$, for the cases in which $10 \sim 40\%$ of nodes are malicious.

It is worth pointing out that the redundant use of the shared keys between a source and each intermediate nodes and the destination node may result in a scalability problem. Generally, in order to not only detect the collusion of $n$ compromised nodes that are consecutively 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. For example, if there are $n$ nodes along a route, then the shared 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.

## V. Simulation Results

In this section, we simulate the proposed TQOS protocol by using the NS-2 simulator [20]. We also compare TQOS to AODV and ARAN by various performance metrics. The simulation parameters and values are shown in Table II, including a small and a large size network.

### A. Network Configurations

In our first scenario, we compare TQOS to the popular AODV by using the large size network with 20 source-destination pairs. The performance metric is simply the total number of packets the protocol delivered at different time. The simulation is run for 700 seconds. It is assumed that the network has a number of malicious nodes, such as 5, 10, or 20. A malicious node randomly drops data packets it is forwarding with a probability of $0.2 \sim 0.5$. We also assume that only intermediate nodes can become malicious during the simulation. For example, along a selected route, if $I_j$ drops packets, then $I_{j+1}$ can detect the dropping and thus update its trustworthiness about $I_j$. In order to simulate the scenario of collusion, as a demonstration purpose, we assume that a normal node updates its trustworthiness with a probability of $70 \sim 90\%$ when it detects a dropping, i.e., some dropping events are detected but not counted in the updating. Therefore, $I_{j+1}$ knows that $I_j$ is a malicious node but does not always report, i.e., they are colluding.

In our second scenario, we compare TQOS to ARAN in the routing performance by using both sizes of networks. Each source-destination pair generates 1000 data packets that need to be delivered. As we discussed in Section IV-A, the security features in TQOS are similar to those in ARAN [10], which have been well simulated and compared to other protocols. Thus, here our focus is the routing performance.

We assume that both TQOS and ARAN use the 512 bit RSA key and 16 byte signature for the public key based security primitives, including digital signature and encryption. In order to compare the performance of TQOS to that of ARAN, we add a delay of 2.2 $ms$ to the processing time of each routing packet in TQOS, same as in ARAN. For data packets, we assume that TQOS uses a 128 bit AES key for the shared key based security primitives. Note that in our simulation, the maximum size data needs to be encrypted is less than $1Kb$, which takes about 1.6276 $\mu s$ to conduct one message verification, for a typical implementation of the shared key operations, e.g., 600 Mbps. Therefore, the time is ignored in our simulation.

In order to compare to ARAN, we also compute the following metrics: packet delivery fraction, average path length, routing load, routing acquisition delay, and end-to-end delay of data packets, as defined in [6].

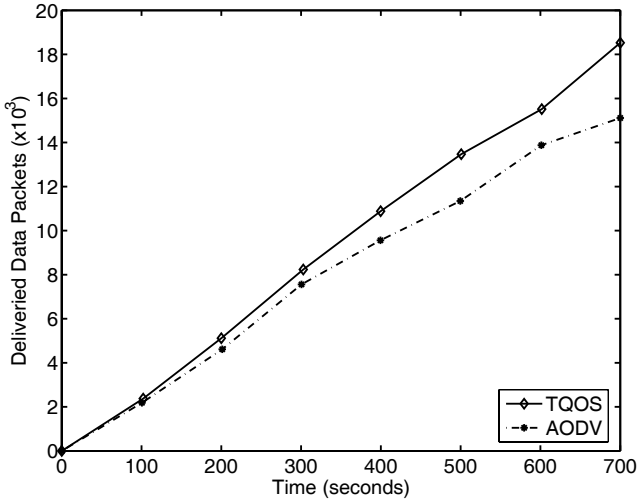


Fig. 2. The total number of packet delivered in the presence of 5 malicious nodes in a network with 50 nodes

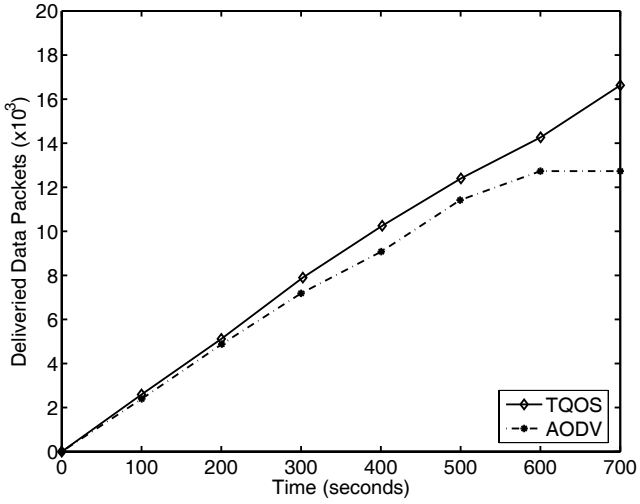


Fig. 3. The total number of packet delivered in the presence of 10 malicious nodes in a network with 50 nodes

### B. Simulation Results

For the first scenario, the results on the total number of delivered packets are shown in Figs. 2, 3, and 4, for the cases of 5, 10, and 20 malicious nodes, respectively. It can be seen that AODV delivers about 15% less of packets than TQOS in the end of the simulation, if only 10% of the nodes are malicious. The reason is that there exists a large number of good nodes between the source and destination. As the percentage of malicious nodes increases to 20% or 40%, TQOS delivers about 10% less of packets in the end of simulation, but AODV stops delivering at time 600 and 500s, respectively. The reason is that AODV has to choose some malicious nodes in forwarding packets if a relatively large number of them are malicious. If AODV does not receive an acknowledgement, it times out and reinitiates a route discovery. But with a high probability, the newly discovered route still contains malicious nodes and thus fails again.

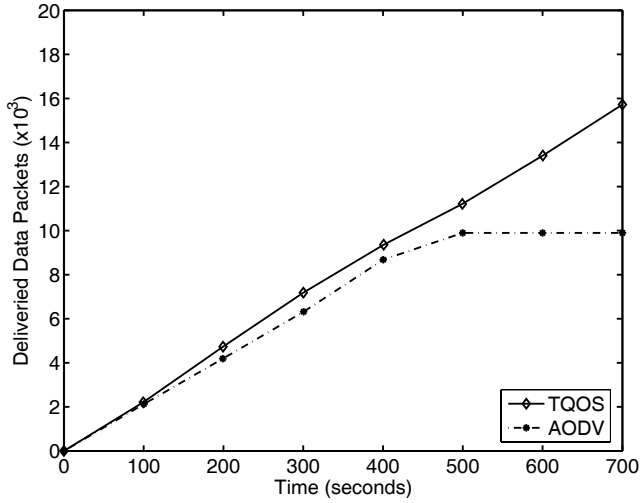


Fig. 4. The total number of packet delivered in the presence of 20 malicious nodes in a network with 50 nodes

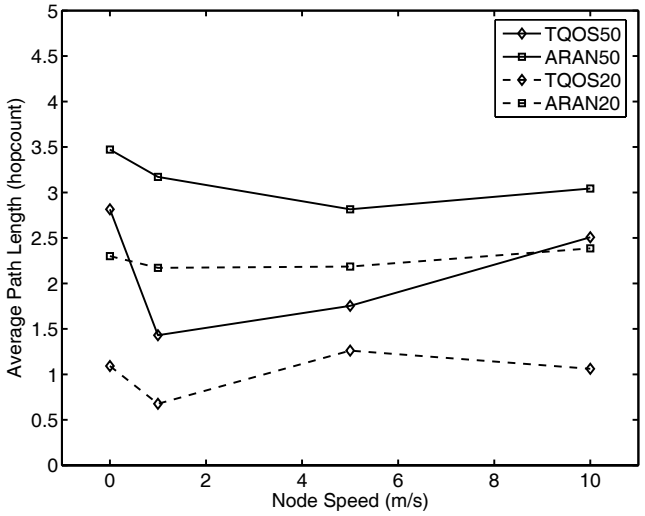


Fig. 6. Average path length of the protocols in two network sizes

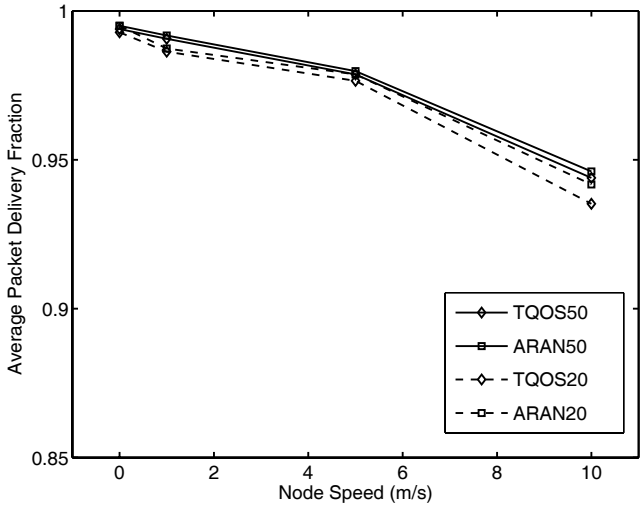


Fig. 5. Packet delivery fraction of the protocols in two network sizes

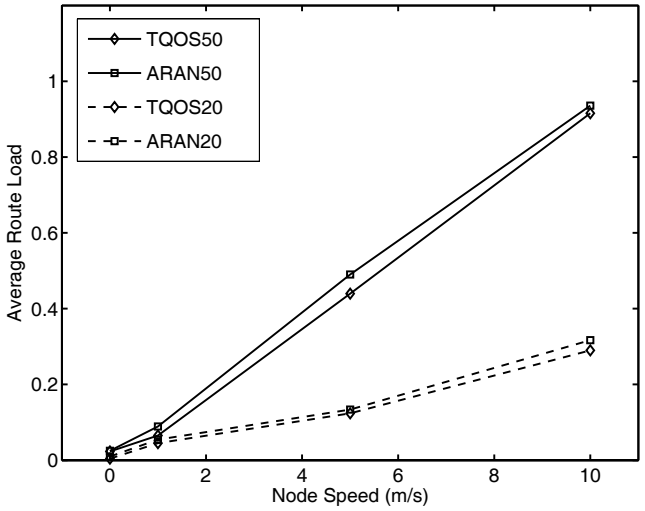


Fig. 7. Average routing load (bytes) of the protocols in two network sizes

On the other hand, TQOS is able to discover routes that go around the malicious nodes and thus keeps delivering packets in both cases. Clearly, TQOS can detect the attacks and maintains a secure route between the source and destination and therefore delivers packets successfully.

For the second scenario, the five performance metrics for TQOS are shown in Figs. 5 $\sim$ 9 and compared to the results of ARAN reported in [10].

Figure 5 compares the packet delivery fraction (PDF) of TQOS to ARAN. Both TQOS and ARAN have a PDF of $94\%$ or higher for both network sizes. In the worst case, in which there are $50$ nodes with a speed of $10\ m/s$, TQOS has a PDF only $0.4\%$ less than ARAN. Therefore, TQOS works as effective as ARAN in discovering and maintaining routes for delivery of data packets.

As shown in Fig. 6, the average path length (APL) for TQOS is always less than that for ARAN in different network sizes and nodal mobility. One reason is that ARAN chooses a route as its best route on which an RDP packet reaches the destination first. By doing this, ARAN may choose a route which has the longest propagation delay but relatively less packet processing time, since the processing time takes a major part in the time the RDP packet being delivered to the destination. While in TQOS, the link quality and traffic delay are parts of the routing metrics. TQOS can always choose a path with the minimum delay to send back a reply if the nodes are of the same trustworthiness, as simulated in this scenario, in which nodal misbehaviors are not assumed. It can be seen that for the network of $20$ nodes with a nodal speed is $1\ m/s$, TQOS has an APL that is less than half of that of ARAN.

Figure 7 shows the average routing load (ARL) measured in bytes. The number of control packets sent in TQOS is almost identical to the one in ARAN. Note that ARAN prohibits an intermediate node from sending reply back to a source, even if it has a route to destination. But TQOS allows the intermediate node to reply, if it is trusted and has the route, because TQOS can assure the security by using trustworthiness. In this way, TQOS avoids sending a lot of control packets. Otherwise, TQOS may have a higher routing load than ARAN due to its use of multiple paths.

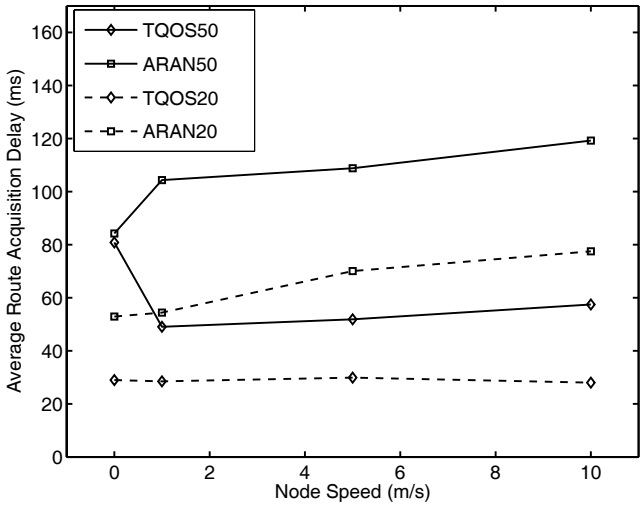


Fig. 8. Route acquisition delay of the protocols in two network sizes

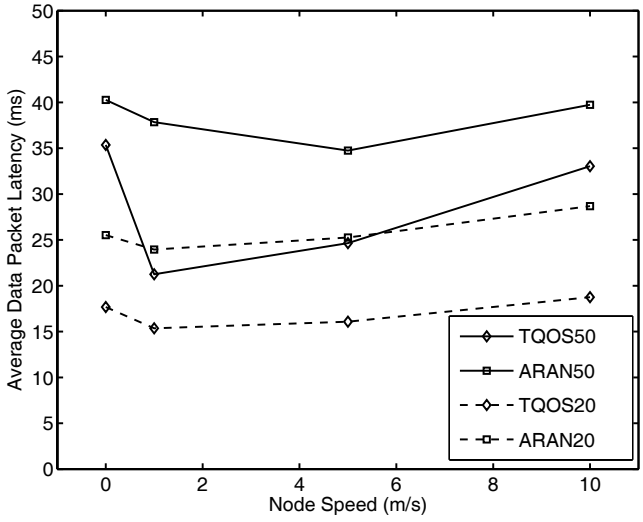


Fig. 9. End-to-end delay of data packets of the protocols in two network sizes

In Fig. 8, it can be seen that TQOS has an average routing acquisition delay (ARAD) about half of the one for ARAN. The major savings on ARAD are due to the fact that TQOS uses more efficient security primitives than ARAN. In ARAN, when a source needs to send out a control packet, it uses RSA to encrypt the packet. Any node that receives the packet needs to verify the digital signature and replace it with its own digital signature. The cryptographic delay is added at each hop, thus the ARAD is increased. In TQOS, if the source has a shared key with its destination, it uses the shared key to encrypt the packet, which almost causes no delay. Same for the intermediate node with its neighbors. Note that the RSA is used to encrypt topology discovery and replay packets only when the source needs to establish a shared key with its destination.

As shown in Fig. 9, the end-to-end delay of data packets in TQOS is always less than the one in ARAN. The reason is that TQOS always tries to choose a route that meets the traffic delay requirement, in addition to its smaller routing acquisition delay than ARAN, as shown in Fig. 8.

In summary, TQOS outperforms the existing protocols including AODV and ARAN.

## VI. Conclusions and Discussions

In this paper, we address the most challenging problem of designing a secure routing protocol with QoS support. For a routing protocol to detect the major internal attacks such as Byzantine attacks, we propose to use both route and message redundancies during topology discovery. The attacks can be detected by verifying various copies of a received message, which reaches a node via different paths at different times. By combining the security mechanism with QoS requirements, we present a secure QoS routing protocol that achieves better performance than the existing ones, as demonstrated by simulation results.

It is worth pointing out that the message redundancy is enforced by sending a same message a few times if the route redundancy does not exist. Also, it is assumed that the source and its intended destination are trusted, which are established by an external trust agent or CA.

It is noted that there are many issues are not addressed in this paper. For example, the procedures for each node to implement and maintain a local certificate repository, the buildup of trust among a node and its neighbors, and the establishment of a self-organized PKI, all need to be further investigated in our future work.

## Acknowledgement

The authors would like to thank the editors and reviewers for their constructional comments and suggestions.

## References

[1] Y. Zhou and Y. Fang, "Scalable and deterministic key agreement for large scale networks," *IEEE Trans. Wireless Commun.*, vol. 6, no. 12, pp. 4366-4373, Dec. 2007.

[2] W. Liu, W. Lou, and Y. Fang, "An efficient quality of service routing algorithm for delay-sensitive applications," *Computer Networks*, vol. 47, no. 1, pp. 87-104, 2005.

[3] P. P. C. Lee, V. Misra, D. Rubenstein, "Distributed algorithms for secure multipath routing in attack-resistant networks," *IEEE/ACM Trans. Networking*, vol. 15, no. 6, pp. 1490-1501, Dec. 2007.

[4] R. Perlman, "Routing with Byzantine robustness," report no: TR-2005-146 [Online]. Available: http://research.sun.com/techrep/2005/smli_tr-2005-146.pdf, Sept. 2005.

[5] C. E. Perkins and E. M. Royer, "Ad hoc on-demand distance vector routing," *Ad Hoc Networking*, Chapt. 5. Addison-Wesley, 2001.

[6] C. E. Perkins, E. M. Royers, and S. R. Das, "Performance comparison of two on-demand routing protocols for ad hoc networks," *IEEE Personal Commun.*, vol. 8, no. 1, p. 16-28, Feb. 2001.

[7] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: a secure on-demand routing protocol for ad hoc networks," in *Proc. 8th ACM International Conf. Mobile Computing Networking*, Sept. 2002.

[8] Perrig, R. Canetti, D. Song, and J. D. Tygar, "Efficient and secure source authentication for multicast," in *Proc. Network Distributed System Security Symposium (NDSS'01)*, pp. 35-46, Feb. 2001.

[9] 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 Computing Syst. Applications*, June 2002.

[10] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. Belding-Royer, "Authenticated routing for ad hoc networks," *IEEE J. Select. Areas Commun.*, vol. 2, no. 1, Mar. 2005.

[11] B. Awerbuch, R. Curtmola, D. Holmer, and C. Nita-Rotaru, "ODSBR: an on-demand secure Byzantine routing protocol," JHU CS Technical Report v.1, Oct. 15th, 2003.

[12] D. Boneh, C. Gentry, H. Shacham, and B. Lynn, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proc. Advances in Cryptology - Eurocrypt03*, LNCS, 2003.

[13] I. Avramopoulos, H. Kobayashi, R. Wang, and A. Krishnamurthy, "Highly secure and efficient routing," in *Proc. IEEE INFOCOM*, Hong Kong, Mar. 2004.
[14] S. Capkun, L. Buttyan, and J. Hubaux, "Self-organized public-key management for mobile ad hoc networks," *IEEE Trans. Mobile Computing*, vol. 2, no. 1, pp. 1-13, Jan./Mar. 2003.
[15] J. Salido, L. Lazos, and R. Poovendran, "Energy and bandwidth-efficient key distribution in wireless ad hoc networks: a cross-layer approach," *IEEE/ACM Trans. Networking*, vol. 15, no. 6, pp. 1527-1540, Dec. 2007.
[16] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures," *Ad Hoc Networks*, pp. 293-315, vol. 1, 2003.
[17] J. Dowling, E. Curran, R. Cunningham, and V. Cahil, "Using feedback in collaborative reinforcement learning to adaptively optimize MANET routing," *IEEE Trans. SMC, Part A: Systems Humans*, pp. 360-372, vol. 35, no. 3, May 2005.
[18] D. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proc. IEEE/ACM MOBICOM*, Sept. 2003.
[19] D. Bertsekas and R. Gallary, *Data Networks*, 2nd ed., Section 5.6 and 5.7. Prentice-Hall, 1992.
[20] The Network Simulator, ns-2 [Online]. Available: http://www.isi.edu/nsnam/ns/.
[21] 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. 2001 IEEE Int. Symp. Circuits Systems (ISCAS'01)*, vol. 4, pp. 650-653, May 2001.
[22] C. Zhang, M. C. Zhou, and M. Yu, "Ad hoc network security: a review," *Int. J. Commun. Syst.*, vol. 20, no. 8, pp. 909-925, Aug. 2007.
[23] M. Yu, S. Kulkarni, and P. Lau, "A new secure routing protocol to defend Byzantine attacks for ad hoc networks," in *Proc. IEEE Int. Conf. Networks (ICON'05)*, vol. 2, pp. 1126-1131, Nov. 2005, Kuala Lumpur, Malaysia.
[24] M. Yu, A. Malvankar, and W. Su, "A distributed radio channel allocation scheme for WLANs with multiple data rates," *IEEE Trans. Commun.*, vol. 56, no. 3, Mar. 2008.
[25] B. Krishnamachari and S. Iyengar, "Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks," *IEEE Trans. Computers*, vol. 53, no. 3, pp. 241-250, Mar. 2004.
[26] L. L. Peterson and B. S. Davie, *Computer Networks: A Systems Approach*, Ch. 8, 3rd ed. Morgan Kaufmann, May 2003.
[27] D. Boneh and M. Franklin, "Identity based encryption from the Weil pairing," *SIAM J. Computing*, vol. 32, no. 3, pp. 586-615, 2003.
[28] U. Kremer, J. Hicks, and J. M. Rehg, "A compilation framework for power and energy management on mobile computers," Dept. CS Technical Report, DCS-TR-446, Rutgers University, June 2001.

**Ming Yu** (M'97-SM'03) received his Ph.D. from Rutgers University, New Brunswick, NJ, in 2002, and a Doctor of Engineering from Tsinghua University, Beijing, in 1994, both in Electrical and Computer Engineering.

He joined AT&T, Middletown, NJ, in July 1997, as a Senior Technical Staff Member. Since 1999, he was with the Dept. of ATM Network Services, AT&T Labs, Middletown, NJ. From December 2002, he worked for the Dept. of IP/Data Network Management System Engineering, AT&T Labs. During Sept. 2003 and Aug. 2006, he was with the Dept. of Electrical and Computer Engineering, State University of New York at Binghamton, NY. As of Sept. 2006, he joined the Dept. of Electrical and Computer Engineering, Florida State University, Tallahassee, FL, as an assistant professor.

His research interests are routing protocols, MAC, QoS, security, energy-efficiency, clustering, radio resource management, traffic engineering, and performance analysis, for both wired and wireless networks. Dr. Yu was awarded an IEEE Third Millennium Medal on May 2000.

**Kin K. Leung** (S'78-M'86-SM'93-F'01) received his B.S. degree from the Chinese University of Hong Kong in 1980, and his M.S. and Ph.D. degrees in computer science from University of California, Los Angeles, in 1982 and 1985, respectively.

He started his career at AT&T Bell Labs in 1986. Following Lucent Technologies spun off from AT&T in 1996, he was with AT&T Labs from 1996 to 2002. In 2002, he re-joined Bell Labs of Lucent Technologies. Since 2004, he has been the Tanaka Chair Professor in Internet Technology at Imperial College, London, UK. His research interests include radio resource allocation, MAC protocol, TCP/IP protocol, mobility management, network architecture, real-time applications and teletraffic issues for broadband wireless networks. He is also interested in a wide variety of wireless technologies, including 802.11, 802.16, and 3G and future generation wireless networks.

He received the Distinguished Member of Technical Staff Award from AT&T Bell Labs in 1994, and was a co-recipient of the 1997 Lanchester Prize Honorable Mention Award. He holds the Royal Society Wolfson Research Merit Award from 2004 to 2009. He is an IEEE fellow. He has published widely and acquired patents in many areas of communication networks. He has actively served on conference committees, including as the committee co-chair for the Multiaccess, Mobility and Teletraffic for Wireless Communications (MMT'98) and the committee Vice-Chair for the IEEE ICC 2002. He was a guest editor for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (JSAC), the MONET journal and the WIRELESS COMMUNICATIONS AND MOBILE COMPUTING journal, and as an editor for the JSAC: Wireless Series. Currently, he is an editor for the IEEE TRANSACTIONS ON COMMUNICATIONS and IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS.