DIMACS Technical Report 2007-17
December 2007

Cryptographic vs. Trust-based Methods for MANET Routing Security

by

Jared Cordasco[1]
Dept. of Computer Science
Stevens Institute of Technology
Hoboken, New Jersey 07030

Susanne Wetzel[2]
Dept. of Computer Science
Stevens Institute of Technology
Hoboken, New Jersey 07030

[1]Affiliated Graduate Student Member
[2]Permanent Member

ABSTRACT

Mobile Ad-hoc Networks (MANETs) allow wireless nodes to form a network without requiring a fixed infrastructure. Early routing protocols for MANETs failed to take security issues into account. Subsequent proposals used strong cryptographic methods to secure the routing information. In the process, however, these protocols created new avenues for denial of service (DoS). Consequently, the trade-off between security strength and DoS vulnerability has emerged as an area requiring further investigation. It is believed that different trust methods can be used to develop protocols at various levels in this trade-off. To gain a handle on this exchange, real world testing that evaluates the cost of existing proposals is necessary. Without this, future protocol design is mere speculation. In this paper, we give the first comparison of SAODV and TAODV, two MANET routing protocols, which address routing security through cryptographic and trust-based means respectively. We provide performance comparisons on actual resource-limited hardware. Finally, we discuss design decisions for future routing protocols.

1 Introduction

In traditional wireless networks, a base station or access point facilitates communications between nodes on the network and communications with destinations outside the network. In contrast, MANETs allow for the formation of a network without requiring a fixed infrastructure. These networks only require that nodes have interoperable radio hardware and are using the same routing protocol to route traffic over the network. The lessened requirements for such networks, along with the ability to implement them using small, resource-limited devices has made them increasingly popular in all types of application areas. For example, MANET-based sensor networks have been proposed to assist in collecting data on the battlefield.

Since there is no fixed infrastructure, the nodes in the network forward traffic for one another in order to allow communication between nodes that are not within physical radio range. Nodes must also be able to change how they forward data over the network as individual nodes move around and acquire or lose neighbors, i.e., nodes within radio range. Routing protocols are used to determine how to forward the data as well as how to adapt to topology changes resulting from mobility.

Initial MANET routing protocols, such as AODV [27], were not designed to withstand malicious nodes within the network or outside attackers with malicious intent. Subsequent protocols and protocol extensions have been proposed to address the issue of security [1, 2, 11, 20, 29, 34, 35, 36]. Many of these protocols seek to apply cryptographic methods to the existing protocols in order to secure the information in the routing packets. It was quickly discovered, however, that while such an approach does indeed prevent tampering with the routing information, it also allows for a very simple denial of service (DoS) attack [15]. This is due to the fact that these protocols require a node to perform expensive cryptographic operations, such as verifying a digital signature, prior to performing any processing on the packet. Therefore, in order to mount a DoS attack, an attacker sends packets with incorrect cryptographic information. All nodes obeying the protocol will expend their processing capabilities attempting to perform the expensive cryptographic operations. This attack is very effective in MANETs as the devices often have limited battery power in addition to the limited computational power. Consequently, this type of DoS attack allows for an attacker to effectively shutdown nodes or otherwise disrupt the network.

The trade-off between strong cryptographic security and DoS has become increasingly important as MANET applications are developed which require a protocol with reasonable security and reasonable resistance to DoS, a kind of middle ground. It has been suggested that various trust mechanisms could be used to develop new protocols with unique security assurances at different levels in this trade-off [6, 37]. However, the arguments for this have been purely theoretical or simulation-based. Determining the actual span of this trade-off in real world implementations is of utmost importance in directing future research and protocol design.

It is in this context that this paper considers two proposed protocol extensions to secure MANET routing. The first, SAODV [35], uses crytographic methods to secure the routing information in the AODV protocol. The second, TAODV [21], uses trust metrics to allow for better routing decisions and penalize uncooperative nodes. While some applications may be able to accept SAODV's vulnerability to DoS or TAODV's weak preventative security, most will require an intermediate protocol tailored to the specific point on the DoS/security trade-off that fits the application. The tailored protocols for these applications will also require performance that falls between that of SAODV and TAODV. Understanding how the SAODV and TAODV protocols (which are on the boundaries of the DoS/security trade-off) perform on real hardware, and to what extent there exists a performance gap is a prerequisite for being able to develop the intermediate protocols. Such evaluation is not only required for developing intermediate protocols, but also for determining the direction for development of new trust metrics for ad-hoc networks.

In this paper we first detail our implementation of these protocols. We then provide the first performance evaluations for these protocols on real world hardware. Based on these results, we then discuss where future efforts should be focused for creating viable routing protocols for real world MANETs.

The remainder of the paper is structured as follows: Section 2 discusses the related work. Overviews of the AODV, SAODV, and TAODV protocols are provided in Section 3. Our experimental setup including hardware, software, and implementation is described in Section 4. Section 5 contains the results of our tests along with a discussion of the results. Conclusions and future work are presented in Section 6.

2 Related Work

Several different protocols have been proposed for ad-hoc routing. The earliest protocols such as DSDV [26], DSR [16], and AODV [27] focused on problems that mobility presented to the accurate determination of routing information. DSDV is a proactive protocol requiring periodic updates of all the routing information. In contrast, DSR and AODV are reactive protocols, only used when new destinations are sought, a route breaks, or a route is no longer in use.

As more applications were developed to take advantage of the unique properties of ad-hoc networks, it soon became obvious that security of routing information was an issue not addressed in the existing protocols. In [19], Lundberg presents several potential problems including node compromise, computational overload attacks, energy consumption attacks, and black hole attacks. Deng et al. further discuss energy consumption and black hole attacks along with impersonation and routing information disclosure in [10]. Jakobsson et al. categorize attacks as manipulation of routing information and exhaustive power consumption, and provide detailed treatments of many characteristic attacks in [15].

Many new routing protocols and extensions to existing protocols have been proposed to address these issues. Due to the extra information available in DSR, by way of source routing, numerous new security protocols are based on it. In [20], Marti et al. extend DSR by adding "watchdog" and "pathrater" mechanisms. One disadvantage of this protocol is that it merely avoids routing through malicious nodes, and it does not do anything to penalize them. This allows a lazy node to not forward traffic for its neighbors while its neighbors will continue to forward its traffic. The protocol proposed by Awerbuch et al. in [1] suffers the same weakness. In [2], Buchegger and Boudee extend DSR in a similar fashion, but address this issue.

Papadimitratos and Haas propose a protocol based on DSR that provides endpoint security by way of pairwise pre-shared symmetric keys in [25]. This, however, does not address security of the routing information over intermediate nodes. In [13], Hu et al. propose Ariadne, a protocol allowing varying levels of computational intensiveness by using either TESLA [28] (based on one-way hash chains), pairwise symmetric keys, or digital signatures.

Several other protocols have been developed which look to minimize computational cost. In [34], Yi et al. introduce a protocol which uses symmetric keys, but instead of using pairwise keys, they use one key per "trust level," a determination that is made when the system is initialized and cannot be changed. Perrig et al. propose SPINS, which uses μTESLA, a lightweight version of TESLA, in [29]. Their protocol, however, assumes the existence of a base station for key setup. In [11], Hu et al. propose SEAD which also uses one-way hash chains. SEAD, however, is a proactive protocol based on DSDV and therefore uses more bandwidth for routing updates than the reactive protocols.

While much research has focused on "lightweight" security mechanisms, some proposed protocols use more expensive asymmetric cryptography. In [36], Zhou and Haas present a multi-path protocol extension that uses threshold cryptography to implement the key management system. It requires some nodes to function as servers and an authority to initialize these servers. Capkun et al. have proposed two protocols involving asymmetric cryptography. In [3], they propose a certificate-based protocol which requires each user to generate their own certificate repository before participating in the network. The protocol they introduce in [4] uses cryptographic methods to derive a node's identity (address) from its certificate. This protocol, however, requires either secure point-to-point side channels or a central authority. Finally, Zapata and Asokan propose SAODV [35], a secure version of AODV, which uses digital signatures and hash chains to secure the routing messages. SAODV is discussed in more detail in Section 3.2.

Applying trust mechanisms to ad-hoc wireless networks has not been given as much attention as the application of cryptographic methods. In addition, trust can be applied to several different areas from informing routing decisions for reliability, to access control, to information flow security. One common thread, however, is that the unreliability of the wireless medium lends itself to Jøsang's subjective logic [17] which takes into account uncertainty. Another common thread is that small world concepts can be applied due to the limited localized connectivity of these networks [9].

Eschenauer et al. provide a thorough summary of the requirements for using trust in ad-hoc networks in [6]. One notable observation is the need to secure the trust information being exchanged from compromise. Zouridaki et al. also provide an evaluation of the requirements for trust in ad-hoc networks in [37]. The authors also detail the problems unique to distance vector algorithms that are absent in source routed algorithms.

In [14], Hughes et al. propose Dynamic Trust-based Resources (DyTR), which uses trust evaluation as a method of access control to network resources. The authors, however, do not discuss the securing of the trust information exchange. Virendra and Upadhyaya also use trust for access control in [33]. Their protocol aims to provide information security for the data traversing the network. Once again, the trust information is unsecured. Nekkanti et al. introduce a routing protocol in [22] aimed at securing information flow. Using security levels on each node and varying the encryption strength accordingly they limit information to flow only over nodes with sufficient clearance. Their proposal does not allow for the security levels to change over the run of the protocol.

Pirzada and McDonald develop a protocol based on DSR in [30]. Their protocol takes advantage of the full route information available in DSR. Unlike other recommendations, however, they only consider trust from direct observations rather than including third party opinions. In [5], Dewan and Dasgupta introduce a protocol to guide routing decisions. In their protocol, however, lazy nodes are not penalized and therefore have no incentive to participate.

In [31], Pissinou et al. propose a trust-based version of AODV using static trust levels. The same authors then extend this protocol in [8] to thwart multiple colluding nodes. Neither of these address securing the trust exchanges, or the overhead involved. Li et al. introduce a trust-based variant of AODV in [18] that secures the trust information. However, their protocol requires an intrusion detection system in the network. Finally, Meka et al. propose a third trusted AODV with a simple method of evaluating trust even without source routing [21]. We describe their protocol in more detail in Section 3.3.

Our work in this paper considers the asymmetric cryptography and trust-based extensions to AODV presented in [35] and [21] respectively and shows a real world comparison of the performance of the two protocols. Our results suggest that new protocols can be developed which take advantage of the best features of both types of protocols, and which share aspects of each security model.

3 Protocol Overviews

In this section we give general overviews of the three protocols we use in our experimental comparisons.

3.1 AODV

AODV is a reactive routing protocol for ad-hoc networks. As such, it only maintains routes between nodes when it is necessary for them to communicate. Unlike other reactive protocols, particularly source routed protocols like DSR, the nodes in an AODV network are only aware of the next hop along a route rather than the entire route itself. This makes AODV harder to secure since the source does not have knowledge of all the nodes participating in the route.

To establish a route to a particular destination D, the source node S broadcasts a Route Request packet (`RREQ`) to its neighbors. S specifies a destination sequence number in the `RREQ` which indicates how "fresh" a route it requires. The `RREQ` is rebroadcast until it either reaches a node with a sequence number for D that is greater than or equal to the one in the `RREQ`, or it reaches D. Intermediate nodes must respond to `RREQ`s if they have a "fresh" enough route as this speeds up route discovery.

As the `RREQ` is forwarded, each node makes a record of the `RREQ` to avoid resending the packet the next time it encounters it, as well as to prepare a reverse route. When the Route Reply (`RREP`) is generated at some intermediate node, or D, it is sent back to S unicast via this reverse route. This sets up the route bi-directionally between S and D. A `RREP` acknowledgement (`RREP-ACK`) can then be sent from S to D.

In addition to the `RREQ`, `RREP`, and `RREP-ACK` messages, `HELLO` messages are used for link-state monitoring. Each node periodically broadcasts a `HELLO` message to inform its neighbors that it is within range of direct radio broadcasts. After missing a certain number of `HELLO` messages from a neighbor, a node sends Route Error (`RERR`) messages to all its neighbors who had established routes forwarding data through the missing node.

3.2 SAODV

AODV was designed without any inherent security mechanisms. Zapata and Asokan try to remedy this by introducing SAODV [35]. The new protocol attempts to prevent impersonation attacks, worm hole and black hole attacks, and denial of service attacks made possible by the use of sequence numbers. SAODV assumes that there is some key management system available that provides public keys for each node in the network, and that individual nodes can securely verify the link between identity and public key for other nodes.

SAODV uses two methods to secure the routing messages. The first method, one-way hash chains, is used to secure the mutable parts of the routing messages. The only mutable fields are the hop count in both the `RREQ` and `RREP` messages. Using a one-way hash chain prevents a node from decrementing the hop count field and is meant to protect against worm hole attacks. To implement this, three new fields are added to the `RREQ` and `RREP` messages, namely, Max_Hop_Count, $Hash$, and Top_Hash. When the message is created, S chooses a random seed and sets $Hash = seed$. It then sets Max_Hop_Count to the time to live value from the IP header. Finally, it sets $Top\_Hash = h^{Max\_Hop\_Count}(seed)$ where h is a hash function and $h^i(x)$ is the iterative hashing of x, i times. Upon receiving a `RREQ` or `RREP`, the receiving node has to confirm that

$$Top\_Hash \overset{?}{=} h^{Max\_Hop\_Count-Hop\_Count}(Hash)$$

It must then increment the Hop_Count field and set $Hash = h(Hash)$.

The second method to protect data in SAODV messages is digital signatures. This is used to prevent impersonation attacks. Digital signatures are used to protect the non-mutable fields of the conventional AODV routing packet along with the non-mutable fields of the SAODV extension. This means that everything is signed except for the Hop_Count and $Hash$ fields. The digital signature along with the public key necessary to verify it are included in the `RREQ`/`RREP` packet. When a node receives one of these messages it is required to verify the signature using the public key provided before it does any further processing.

If the signature fails to verify, the packet is discarded. If the signature is verifiable then the node proceeds as per AODV.

As per the SAODV specification, the longest key length that can be used for the digital signatures is only 512 bits. The are two main issues with using stronger key lengths. The first is obviously the computational overhead required for the operations with a larger key length. The other issue is more important in ad-hoc networks than in other settings where the communication medium is more reliable. The problem is the increased message size. For example, with SAODV's most basic signature method, the Single Signature Extension (SSE), and a 512 bit RSA key, a 24 byte `RREQ` message is extended to 198 bytes and the 20 byte `RREP` and `HELLO` messages become 194 bytes in length.

Aside from the transmission and computational overhead of digital signatures, there is one other problem with signing the AODV messages. When a signed `RREQ` reaches the destination, the destination sends back a signed `RREP`. This signature ensures that the route is authentic in that the `RREP` is truly from the *destination*. However, in conventional AODV, intermediate nodes are supposed to send `RREP` messages if they have a fresh enough route. In SAODV, the intermediate nodes cannot sign the `RREP` as though it came from the destination. Therefore, SAODV has two digital signature methods.

In the first, a Single Signature Extension (SSE) is used and only the destination can respond to `RREQ` messages. In the second, when the destination responds to a `RREQ`, it includes two different signature it has generated. The first is the signature that would be included in a SSE `RREP`. The second is an extra signature which the intermediate node can use to respond to `RREQ`s it receives thereafter. This is called the Double Signature Extension (DSE). It should be noted that while the `RREP` with DSE contains a lifetime set by the destination, the latest draft standard is unclear about exactly how this prevents a malicious node from reusing a validly obtained DSE for malicious purposes.

Since `HELLO` messages are `RREP` messages with a time to live of 1, they are secured in the same way as regular `RREP` messages generated in response to `RREQ`s. `RERR` messages, however, are handled differently. Even though there is a large amount of mutable information in the `RERR` packets, the SAODV protocol requires that the entire packet be signed so that the receiver can verify the sender is who it claims to be. To rebroadcast a `RERR` message, a node must repackage it with its own signature adding or updating any information in the process. In addition, to prevent denial of service attacks caused by artificially inflated sequence numbers, nodes do not update their stored newest sequence number for a particular destination based on the sequence number in a `RERR` message.

SAODV aims to prevent several different types of attacks. Using digital signatures on each routing packet establishes the identity of the source of that packet and the information contained therein. This prevents a node from impersonating another node to disseminate false information. The digital signature also serves to verify that the receiving node does in fact get the values the source node transmitted (for the non-mutable fields) and that no node in between changed these values. The use of one-way hash chains to verify hop count values prevents a single malicious node from advertising false information under its own name. If a node could report any hop count value, it could choose either a low or high value to get either selected or avoided, respectively, for inclusion along the route. This is prevented if there is a single attacker. However, if there are multiple colluding attackers in different areas of the network, they can bypass this mechanism using techniques similar to those involved in the wormhole attack presented in [12].

3.3 TAODV

Many routing protocols use the number of hops from source to destination as the metric for determining the best route. Meka et al. proposed a variant of AODV which instead uses a computed trust value as the metric for routing decisions [21]. Nodes in a TAODV network maintain trust values for their neighbors and the routes they have discovered. We first discuss the modifications to AODV that are necessary. We then describe how these modifications are used to compute trust values.

3.3.1 Modifications:

In order to track trust values for their neighbors, TAODV nodes require a new data structure called the Neighbors' Trust Table (NTT). Each entry in the NTT stores the neighbor's node ID, the trust value for that neighbor and the current number, r, of `RREQ`s this node can send. In addition to adding the NTT, there are modifications necessary to the routing table. The routing table entry for a particular destination is modified to hold *all* the routes from that node to the destination having the highest destination sequence number. Each individual route is assigned a unique route ID, R_{id}, which is stored along with the Advertised Trust Value (ATV) and the computed Route Selection Value (RSV).

Tracking route trust values is accomplished through extensions to the `RREQ`, `RREP`, and `RREP-ACK` packets along with a new packet for congestion control called the `CHOKE` packet. The `RREQ` packet is extended with the `Omit Node Flag` and `Omit Node ID` fields. This allows the source to specify nodes that are to be precluded from being in the route to the destination. All nodes must ensure that they are not forwarding a `RREP` from a neighbor if that neighbor is identified in the `Omit Node ID` field.

The modified `RREP` packet includes fields for `Route Trust` and `Recommender ID`. As a `RREP` returns to the source node, each intermediate node caches the value in `Route Trust`. If the node has not computed a trust value for the this route, it simply forwards the `RREP` to the next upstream node. However, if the node has already computed a trust value for the route to this destination, it places its value in `Route Trust` and its identity in `Recommender ID` before forwarding the packet.

The `RREP-ACK` message is not only modified, but also repurposed. Due to this, it is renamed as `R_ACK`. The `R_ACK` message is a basic `RREP-ACK` with additional fields for `Packets Received` and `Timestamp`. This packet is a reporting packet periodically sent by the destination to inform all nodes along the route, including the source, of the number of packets received since the last `R_ACK`. The information in the `RREP` and `R_ACK` messages is used to compute trust values as described below.

The `CHOKE` packet consists of only three fields: `Node ID`, `Timestamp`, and `Lifetime`. It is broadcast by a node to report to its neighbors that it is currently receiving, sending, or forwarding large amounts of traffic and is therefore congested. By allowing a node to inform its neighbors of congestion, the protocol prevents a node from being penalized for dropping traffic. If the node is still congested after the first `CHOKE` packet's `Lifetime` expires, it can send another `CHOKE` packet. To prevent lazy nodes from abusing this system, a threshold time $t_{congmax}$ is set. If a node reports congestion for a period of time longer than $t_{congmax}$, then its neighbors begin monitoring its traffic. If the node is actually congested, the neighbor tries to find other routes to replace those using the congested neighbor. If the node is not congested, it is penalized for selfish behaviour as described below.

3.3.2 Computation of Trust Values:

Nodes implementing TAODV measure an Observed Trust Value (OTV) for each route in use. This is a straight forward computation of the number of packets received (as reported in the `R_ACK` message) divided by the total number of packets forwarded or sent by the node performing the calculation. The OTV is then compared to the ATV for the route. This is used in an incentive/penalty system to determine how a node updates its trust value for the downstream neighbor. This system uses four system-wide parameters i, p_ℓ, p_h, and p_c which are discussed later. If the OTV is within a threshold r_{thresh}, i.e.,

$$|\text{OTV} - \text{ATV}| \leq r_{thresh}$$

then the neighbor is given incentive $I = r * \left(\frac{i}{H}\right)$, where r is the current number of `RREQ`s the node can send as stored in the NTT and H is the distance of the node being evaluated from the destination in hops. If the neighbor fails to live up to its ATV, namely if OTV is below the threshold, the neighbor is given penalty $P_\ell = -r * \left(\frac{p_\ell}{H}\right)$. Likewise, if the neighbor outperforms its ATV, it must be penalized or else nodes would advertise low ATVs to avoid being selected to forward data. This penalty is $P_h = -r * \left(\frac{p_h}{H}\right)$. Finally, nodes caught broadcasting false `CHOKE` packets are given penalty $P_c = -r * \left(\frac{p_c}{H}\right)$.

After a penalty or incentive is assessed, it must be added to the nodes r in the NTT, thus rewarding or penalizing the node by cooperating more or less frequently on `RREQ`s. For example, if a node n underperforms, then its upstream neighbor updates n's entry in its NTT such that $r = r + P_\ell$. It is important to note that in all of these incentives/penalties, the node's distance from the destination is inversely proportional to the effect of the incentive/penalty. This helps prevent downstream nodes from souring the reputation of upstream nodes. While exact values for i, p_ℓ, p_h, and p_c are not given, the following recommendation is made [21]:

$$0 \leq p_c \leq p_h \leq p_\ell \leq i \leq 1$$

Once NTT trust values are up-to-date, a node must recalculate the RSV stored in the routing table. The following method allows an administrator to adjust the weights to their particular situation:

$$\text{RSV} = \alpha_1 \left(\frac{T_{ind}}{T_{avg}}\right) + \alpha_2 \left(\frac{RT_{ind}}{RT_{avg}}\right) + \alpha_3 \left(\frac{H_{avg}}{H_{ind}}\right)$$

where

T_{ind}	:	Trust in the individual neighbor.
T_{avg}	:	Average trust for all neighbors with routes to this destination.
RT_{ind}	:	Trust in the individual route.
RT_{avg}	:	Average trust for all routes to this destination.
H_{ind}	:	Hopcount for the individual route.
H_{avg}	:	Avg. hopcount for all routes to this destination.

Once again, specific values for the system-wide parameters α_1, α_2, and α_3 are not given. It is suggested that all three parameters be restricted to values between 0 and 1 such that $\alpha_1 + \alpha_2 + \alpha_3 = 1$. and that α_1 and α_2 be much larger than α_3. However, these values can be set to any value at the administrator's discretion as the paper states that any desired method for calculating RSV can be used.

After the new RSV is calculated, the node reconsiders all routes to that particular destination. It selects the route with the highest RSV to be the new active route and updates its routing table accordingly.

Unlike SAODV, TAODV does not aim to prevent specific attacks through its additions to AODV. Instead it looks to detect these attacks and provide the node with sufficient information to re-route data around the attack.

4 Experimental Setup

Since ad-hoc networking's most promising applications make use of small, resource-constrained devices that are significantly different from today's ever faster desktop computers, special attention must be paid to the trade-off between strong cryptographic security and DoS. While theoretical analysis or simulation may give helpful hints on the relative efficiency of different approaches, only real world implementation and performance testing can give a concrete picture of the actual width of this spectrum. Such measurements provide the necessary information to determine which protocols are suitable for specific applications. In addition, the results can then be used to guide the design of novel protocols better suited to particular deployment situations.

In order to get an understanding for the real world performance of the AODV, SAODV, and TAODV protocols, we have implemented each of them on real hardware and measured their performance. In this section we detail the setup for the experiments used to acquire these measurements. We first describe the supporting hardware and software setup for our implementations. We then present the specifics of the actual implementation for each of the three protocols. Finally, we detail the design of the experiments used to evaluate the protocols and explain why these tests are more relevant than other more common metrics.

4.1 Hardware and Software Setup

To gain an accurate grasp of protocol performance, the tests must be run hardware equivalent to that which is commonly used in ad-hoc networks. For our testing we used the Sharp Zaurus

CPU	206MHz Intel SA-1110 StrongARM
Memory	64MB DRAM + 16MB Flash ROM
Card Slots	1 × Compact Flash type II, 1 × Secure Digital
Battery	950 mAH Lithium Ion

Table 1: Zaurus SL-5500 Hardware Specifications

SL-5500 model palmtops. Table 1 lists the hardware specifications for the SL-5500. From the listing in Table 1, we see that the Zaurus is as powerful as a desktop computer was a decade ago. With the rapid advances in technology, a device with these capabilities could become the embedded sensor network device of the near future. Regardless, they allow for an analysis using processors that are an order of magnitude out of step with today's conventional processors.

Each Zaurus was equipped with a Linksys WCF11 compact flash card for wireless communication. The Zauruses ran OpenZaurus [24] v3.5.4, an embedded version of Linux. In order to compile programs for the Zaurus we used a cross-compiler toolchain based on GCC v3.3.4. In addition, as described in Section 4.2, our code requires the OpenSSL [23] libraries. For this purpose, OpenSSL v0.9.7j was cross-compiled and statically linked into executables where necessary. All cross-compiling was performed on a desktop running Slackware Linux 11.0 [32].

4.2 Implementation

Our AODV implementation is the result of previous projects in this area [38]. The implementation is designed to run on the Linux operating system. As with many other AODV implementations for Linux, it separates functionality into a kernel module and a userspace daemon. The kernel module uses hooks in the netfilter interface to send packet headers from the wireless interface to the userspace daemon. The daemon then determines how to handle the packet. If the packet is a routing control packet, then the daemon processes the packet in accordance with the AODV specification. If instead the packet is a data packet, the daemon determines whether or not a route exists to the necessary destination. If there is a suitable route, the packet is flagged and the kernel module queues it to be sent out. If no route exists, the daemon begins route discovery. Once a route is found, the daemon enters the route into the kernel's routing table. It then flags the packet (and any additional packets arriving during discovery) to be queued for transmission. The implementation is written completely in C.

In order to implement SAODV, it was necessary to have a library of cryptographic operations. We used OpenSSL for this purpose, and we developed a security library which wrapped much of OpenSSL's functionality into components appropriate for ad-hoc routing purposes. One particularly useful feature of the security library is that it allows easy use of several different OpenSSL contexts at once. For SAODV, this was useful as nodes must switch between signing, verifying, and hash chain operations rapidly to both send and receive routing messages. New data structures were added for SAODV's single signature extension

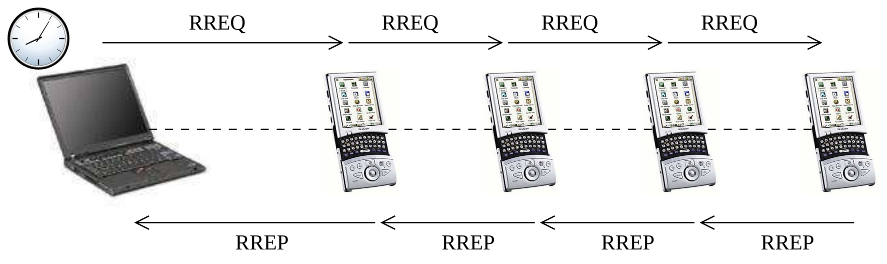

Figure 1: Network Setup for Round Trip Timing Tests

and the necessary code was added to the message processing functions for `RREQ`, `RREP`, `HELLO`, and `RERR` messages. The design of the AODV implementation allowed SAODV functionality to be implemented while maintaining one binary with the ability to run both protocols.

Implementing TAODV required many additions similar to those involved in SAODV. New data structures were used for the NTT as well as the extended messages and the new `R_ACK` message. Similarly, message handling functions were updated to use the extensions and take the appropriate actions. One challenge in implementing TAODV was counting packets sent, forwarded, or received for a particular route. While it intuitively seems to be something that should be implemented in the kernel module that is already tied into the netfilter framework, this would require extra data exchange between the kernel module and the daemon. Since our implementation already passes packet headers to the daemon for route discovery initiation and flagging, it was sufficient to place the counting mechanism in the daemon.

Keeping track of the additional routing information required significant extension of our AODV implementation. The original implementation does not support any multi-path entries in the routing table. Modifying it to support such a setup for TAODV would have required rewriting significant amounts of the base AODV code. Instead, we implemented a multi-path capable routing table for use exclusively by the TAODV protocol. When a node initially discovers a route, or changes the active route to a particular destination, it merely copies the necessary entry to the daemon's local routing table and marks it as having been altered so that it is updated in the kernel's routing table at the next sync. This simplified the implementation using only a negligible amount of extra memory.

4.3 Testing Setup

There were two performance factors we were interested in for the purposes of this comparison. The first is the per-packet processing overhead. It is important to note that only CPU time was measured. Therefore this overhead reflects use of the processor by each protocol. In these tests we use AODV as a baseline. Thus, for SAODV we measure the time it takes to generate an SSE for `RREQ`, `RREP`, and `HELLO` messages. We also measure the time it takes for a node to verify an SSE for those same messages. For TAODV we measure how long it takes a node to generate or process and update `RREP` and `R_ACK` messages. Due to the fact

Operation	Proc. Time (ms)	Std. Dev.
SSE generation	30.8	0.028
SSE validation	3.81	0.006

Table 2: SAODV Per-Packet Overhead Times

that some of the operations we measure have a runtime less than the resolution of our timer (10 ms as per the Linux kernel), we perform a large number of operations back-to-back per measurement. We then make multiple measurements.

Our second performance metric is round trip time for route discovery. The justification for this metric lies in the fact that we are looking at securing the routing control packets. Once a route is established, data is forwarded with the same efficiency regardless of the routing protocol. Therefore, it is important to see how the per-packet overhead along with the increased packet size affect the time for route discovery. For this test, we measure the performance of AODV in addition to that of SAODV and TAODV. This is necessary because both AODV and TAODV will generate `RREP`s after fewer hops when the destination's neighbor responds, while SAODV requires that the destination itself responds. For our experiments, we used a five node network consisting of one laptop and four Zauruses as illustrated in Figure 1. We used the network sniffer ethereal [7] running on the laptop to measure the time elapsed from the sending of the `RREQ` to the receipt of the `RREP`. These individual measurements were also performed repeatedly as explained in Section 5.

5 Results

In this section we present our results for both the per-packet and round trip time tests described in Section 4.

5.1 Per-Packet Results

For the per-packet overhead tests, we measured the amount of processing time a node spends above and beyond that required for conventional AODV. All tests were performed on the Zauruses with only the necessary software running (i.e., no graphical login manager, no X server, etc.). In the SAODV tests, we measure generation and validation of the SSE which requires hash computation and a digital signature/verification. The hash function used for these tests was MD5 and the digital signature/verification was performed using a 512-bit RSA key pair. There were 1000 operations run per measurement and 1000 measurements overall. Table 2 shows the results of our SAODV tests.

Consequently, in order to send a `RREQ`, `RREP`, or `HELLO` message, the node spends 30.8 milliseconds generating the SSE. The significant impact on performance occurs in generating the SSE for `HELLO` messages since they are sent periodically. According the to AODV specification, a node should send a `HELLO` message every `HELLO_INTERVAL` milliseconds unless it

Operation	Proc. Time (ms)	Std. Dev.
RREP/HELLO send	0.0453	0.002
RREP/HELLO processing	0.0452	0.002
R_ACK send	0.193	0.004
R_ACK processing	0.297	0.005

Table 3: TAODV Per-Packet Overhead Times

has broadcast any messages during the previous interval. This means that only RREQ and RERR messages could prevent sending a HELLO message, as all other messages are unicast. Obviously, this can place a significant burden on each node.

Since SAODV requires that each message with a SSE is validated before any further processing takes place, each RREQ and RREP gets delayed 3.8 milliseconds at each hop which forwards it. In addition, HELLO messages take the same amount of time to be validated. While nodes are supposed to let ALLOWED_HELLO_LOSS * HELLO_INTERVAL milliseconds pass before deciding a link is broken and a neighbor should be removed from its routing table, it is conceivable that on a node with several neighbors and a large amount of data to forward, route status may fluctuate for some neighbors whose HELLO packets get delayed in validation.

In TAODV, we measure the per-packet overhead for RREP, HELLO, and R_ACK messages. The system-wide parameters discussed in Section 3.3 do not influence the overhead of TAODV for any of the tests we performed. However, it was necessary to fix these values to allow for the calculation of RSV. For all TAODV tests we used the following system-wide parameter values: $i = 0.8, p_\ell = 0.6, p_h = 0.4, p_c = 0.2, \alpha_1 = 0.4, \alpha_2 = 0.4$, and $\alpha_3 = 0.2$. Due to the very small running time of the operations, one million operations were performed per measurement and 5000 measurements were taken. Table 3 shows the results for the TAODV tests.

As the results show, there is much less per-packet overhead for TAODV when compared to SAODV. The main source of overhead involved the R_ACK packets. Since the R_ACK packets are new packets rather than packet extensions, it was necessary to allocate a packet buffer in the message sending system of our implementation each time a R_ACK packet was to be sent. With other messages that were extended, the packet buffer was already allocated and the extension was simply written into free space at the end. This difference contributed significantly to the 0.193ms overhead for sending the R_ACK message.

The overhead for processing the R_ACK message was almost completely due to the recalculation of the OTV and RSV values. The TAODV implementation used double primitives for all calculations in order to keep with the protocol description in [21]. However, this affects the performance since the SA-1110 processor in the Zaurus has only integer arithmetic units. For systems with less computational power than the Zaurus' these results suggest that it may be necessary to rewrite trust-based metrics into their equivalent using integer arithmetic instead.

Protocol	Round Trip Time (ms)	Std. Dev.
AODV	138.177	0.765
SAODV	324.732	7.22
TAODV	152.780	0.863

Table 4: Round Trip Times

5.2 Round Trip Results

The round trip tests for route discovery were performed for all three protocols. This was particularly important due to the differences in which node sends the `RREP` as described in Section 4.3. Due to the nature of the measurements, only one route discovery operation could be executed per measurement. Overall 5000 of these individual measurements were performed. Table 4 shows the results of the tests.

These results show that SAODV is indeed a significantly more expensive protocol. Specifically, SAODV takes 2.35 times as long as conventional AODV to get a `RREP` back to a `RREQ` originator. This is due, in part, to the added cryptography and increased message size. This is also due to the inability of intermediate nodes to respond to `RREQ`s. Traversing the additional hop in both directions adds to the latency. While we did not implement the DSE, this should not have a large effect on the average route discovery since a destination now has to generate two digital signatures for a `RREP`. In addition, DSE only addresses the overhead incurred by intermediate nodes not responding to `RREQ`s. There still is overhead from the added cryptography and increased message size which implementing DSE will not solve.

The results also show that the use of SAODV will require adjustments to the recommendations for configurable parameters in AODV. This is missing from the current draft standard for SAODV. For example, the current suggested `NODE_TRAVERSAL_TIME` is 40 ms which results in `NET_TRAVERSAL_TIME` being set to 1400 ms. The value of `NET_TRAVERSAL_TIME` serves as the timeout for `RREQ` messages. Consequently, as per the results above, if these parameters were not adjusted, nodes would have problems discovering routes of length greater than seventeen hops. In some applications this may not cause problems. However, in certain applications such as large area sensor networks, routes of this length or greater would not be unreasonable to expect.

TAODV, on the other hand, takes only 1.11 times as long as AODV. This shows that the trust-based calculations and additional information exchange can be used without incurring the overhead of SAODV. While there is some expense for the trust calculations, it is not nearly as expensive as the cryptographic operations. The results show that TAODV is indeed at the opposite end of the trade-off from SAODV. This is due to the fact that the TAODV information itself in each packet is not secured.

Overall, the results show that there is indeed a wide spectrum in the trade-off between cryptographic security and DoS. By adding an appropriate lightweight security mechanism to secure the trust information in the routing packets, a hybrid protocol can be created which is less expensive than SAODV and more secure than TAODV. Future protocol designs should seek to use various new combinations of smarter, trust-based metrics and lightweight security mechanisms in order to develop hybrid protocols across this spectrum.

6 Conclusion

In this paper, we have compared the SAODV and TAODV protocols for securing ad-hoc network routing. We presented the results of implementation and evaluation of both protocols on real resource-limited hardware. The expected difference between the two protocols was shown to be consistent with this real world scenario. These experiments showed that there is significant room between the two protocols for a secure hybrid protocol to be developed which takes advantage of the strongest points of both.

Future work needs to delve further into the extensive body of work on various trust metrics. This includes the testing of other trust metrics for use in ad-hoc routing as well as developing the aforementioned hybrid protocols and testing their performance against the results presented in this paper. In addition, it is necessary to test the quality of the routing decisions produced by all of these protocols in a malicious environment.

References

[1] C. N.-R. Baruch Awerbuch, David Holmer and H. Rubens. An On-Demand Secure Routing Protocol Resilient to Byzantine Failures. In *Proceedings of the ACM Workshop on Wireless Security (WiSe02)*, Atlanta, Georgia, September 2002.

[2] S. Buchegger and J.-Y. L. Boudec. Nodes Bearing Grudges: Towards Routing Security, Fairness, and Robustness in Mobile Ad hoc Networks. In *Proceedings of the Tenth Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 403 – 410, Canary Islands, Spain, January 2002. IEEE Computer Society.

[3] S. Capkun, L. Buttyan, and J. Hubaux. Self-Organized Public-Key Management for Mobile Ad hoc Networks. In *Proceedings of the ACM Workshop on Wireless Security (WiSe02)*, 2002.

[4] S. Capkun, J. Hubaux, and L. Buttyan. Mobility Helps Security in Ad hoc Networks. In *Proceedings of the ACM Symposium on Mobile Ad hoc Networking and Computing (MobiHOC)*, June 2003.

[5] P. Dewan and P. Dasgupta. Trusting Routers and Relays in Ad hoc Networks. In *ICPPW '03: Proceedings of the 2003 International Conference on Parallel Processing Workshops*, pages 351–358, 2003.

[6] L. Eschenauer, V. Gligor, and J. Baras. On Trust Establishment in Mobile Ad-hoc Networks. Technical Report MS 2002-10, Institute for Systems Research, University of Maryland, 2002.

[7] Ethereal - A Network Protocol Analyzer. `http://www.ethereal.com/`.

[8] T. Ghosh, N. Pissinou, and K. Makki. Collaborative Trust-based Secure Routing Against Colluding Malicious Nodes in Multi-hop Ad hoc Networks. In *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04)*, pages 224–231, Washington, DC, USA, 2004. IEEE Computer Society.

[9] E. Gray, J. Seigneur, Y. Chen, and C. Jensen. Trust Propagation in Small Worlds. In *Proceedings of the First International Conference on Trust Management (iTrust2003).*, volume 2692 of *LNCS*, May 2003.

[10] W. L. H. Deng and D. P. Agrawal. Routing Security in Wireless Ad hoc Networks. *IEEE Communications Magazine*, pages 70–75, 2002.

[11] Y. Hu, D. Johnson, and A. Perrig. SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad hoc Networks. *Ad hoc Networks*, I:175–192, 2003.

[12] Y. Hu, A. Perrig, and D. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad hoc Networks. Technical report, Department of Computer Science, Rice University, December 2001.

[13] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad hoc Networks. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 12–23, New York, NY, USA, 2002. ACM Press.

[14] T. Hughes, J. Denny, P. A. Muckelbauer, and J. Etzl. Dynamic Trust Applied to Ad hoc Network Resources. In *Proccedings of the Autonomous Agets and Multi-Agent Systems Conference*, 2003.

[15] M. Jakobsson, S. Wetzel, and B. Yener. Stealth Attacks on Ad hoc Wireless Networks. In *Proceedings of IEEE Vehicular Technology Conference (VTC-Fall 2004)*, 2004.

[16] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad hoc Wireless Networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[17] A. Josang. An Algebra for Assessing Trust in Certification Chains. In *Proceedings of the Network and Distributed Systems Security (NDSS'99) Symposium.* The Internet Society, 1999.

[18] X. Li, M. Lyu, and J. Liu. A Trust Model Based Routing Protocol for Secure Ad hoc Networks. In *Proceedings of the Aerospace Conference*, pages 1286–1295, March 2004.

[19] J. Lundberg. Routing Security in Ad hoc Networks. Technical Report Tik110. 501, Helsinki University of Technology, 2000.

[20] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad hoc Networks. In *Mobile Computing and Networking*, pages 255–265, 2000.

[21] K. Meka, M. Virendra, and S. Upadhyaya. Trust Based Routing Decisions in Mobile Ad-hoc Networks. In *Proceedings of the Workshop on Secure Knowledge Management (SKM 2006)*, 2006.

[22] R. K. Nekkanti and C. wei Lee. Trust based adaptive on demand ad hoc routing protocol. In *ACM-SE 42: Proceedings of the 42nd annual Southeast regional conference*, pages 88–93, New York, NY, USA, 2004. ACM Press.

[23] OpenSSL: The Open Source Toolkit for SSL/TLS. `http://www.openssl.org/`.

[24] OpenZaurus: Opensource Linux Distribution for Sharp Zaurus Palmtops. `http://www.openzaurus.org/`.

[25] P. Papadimitratos and Z. Haas. Secure Routing for Mobile Ad hoc Networks. In *Proceedings of SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, pages 27–31, 2002.

[26] C. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[27] C. E. Perkins and E. M. Royer. Ad-hoc On-Demand Distance Vector Routing. *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications (WMCSA)*, page 90, 1999.

[28] A. Perrig, R. Canetti, D. Tygar, and D. Song. The TESLA Broadcast Authentication Protocol. In *Cryptobytes, Volume 5, No. 2*, pages 2–13. RSA Laboratories, 2002.

[29] A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, and J. D. Tygar. SPINS: Security Protocols for Sensor Networks. In *Mobile Computing and Networking*, pages 189–199, 2001.

[30] A. Pirzada and C. McDonald. Establishing Trust in Pure Ad-hoc Networks. In *Twenty-Seventh Australasian Computer Science Conference (ACSC2004)*, 2004.

[31] N. Pissinou, T. Ghosh, and K. Makki. Collaborative Trust-Based Secure Routing in Multihop Ad hoc Networks. In *NETWORKING 2004, Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications*, pages 1446–1451, 2004.

[32] The Slackware Linux Project. `http://www.slackware.org/`.

[33] M. Virendra and S. Upadhyaya. Securing Information through Trust Management in Wireless Networks. In *Proceedings of the Workshop on Secure Knowledge Management (SKM 2004)*, pages 201–206, 2004.

[34] S. Yi, P. Naldurg, and R. Kravets. Security-Aware Ad hoc Routing for Wireless Networks. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing (MobiHoc'01)*, pages 299–302, New York, NY, USA, 2001. ACM Press.

[35] M. G. Zapata and N. Asokan. Securing Ad hoc Routing Protocols. In *Proceedings of the ACM Workshop on Wireless Security (WiSe02)*, pages 1–10, New York, NY, USA, 2002. ACM Press.

[36] L. Zhou and Z. J. Haas. Securing Ad hoc Networks. *IEEE Network Magazine*, 13(6):24–30, 1999.

[37] C. Zouridaki, B. L. Mark, M. Hejmo, and R. K. Thomas. A Quantitative Trust Establishment Framework for Reliable Data Packet Delivery in MANETs. In *Proceedings of the 3rd ACM workshop on Security of Ad hoc and Sensor Networks (SASN'05)*, pages 1–10, New York, NY, USA, 2005.