TrueLink: A Practical Countermeasure to the Wormhole Attack in Wireless Networks

Jakob Eriksson, Srikanth V. Krishnamurthy, Michalis Faloutsos
University of California, Riverside

***Abstract*— In a wormhole attack, wireless transmissions are recorded at one location and replayed at another, creating a virtual link under attacker control. Proposed countermeasures to this attack use tight clock synchronization, specialized hardware, or overhearing, making them difficult to realize in practice.**

TrueLink is a timing based countermeasure to the wormhole attack. Using TrueLink, a node i can verify the existence of a direct link to an apparent neighbor, j. Verification of a link $i \leftrightarrow j$ operates in two phases. In the rendezvous phase, the nodes exchange nonces α_j and β_i. This is done with tight timing constraints, within which it is impossible for attackers to forward the exchange between distant nodes. In the authentication phase, i and j transmit a signed message (α_j, β_i), mutually authenticating themselves as the originator of their respective nonce.

TrueLink does not rely on precise clock synchronization, GPS coordinates, overhearing, geometric inconsistencies, or statistical methods. It can be implemented using only standard IEEE 802.11 hardware with a minor backwards compatible firmware update. TrueLink is meant to be used together with a secure routing protocol. Such protocols require an authentication mechanism, which will also be used by TrueLink. TrueLink is virtually independent of the routing protocol used. Our performance evaluation shows that TrueLink provides effective protection against potentially devastating wormhole attacks. [1]

I. Introduction

Security is a crucial component of any wireless routing protocol to be used outside the laboratory. A wireless network can be attacked at all layers of the protocol stack. A particularly difficult attack to counter is the wormhole attack. In the wormhole attack [1], [2], [3], an attacker, or potentially multiple colluding attackers, surreptitiously relay packets between distant locations. This can give a node the impression that it is the neighbor of a node that is far away. By faking links between distant nodes, attackers may be able to manipulate nodes to send traffic through them, where the attackers can drop, modify or record such traffic. We define an **attacker node** as a node brought in by the attacker, and a **compromised node** as a node that was once a legitimate member of the network, but has been taken over or infiltrated by the attacker. The wormhole attack is particularly dangerous in that it can be mounted without compromising any nodes, and without any knowledge of the protocols used. Furthermore, the attackers can mount the attack without revealing their identities: the fake edge appears between two distant legitimate nodes.

[1]Prepared partially through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

The wormhole attack was independently introduced in [1], [2] and [3]. Most secure and non-secure ad hoc routing protocols proposed (for example [4], [5], [6], [7]), are vulnerable to the wormhole attack. Previous countermeasures to the wormhole attack have relied on specialized hardware; employing such hardware may not be feasible in typical wireless network scenarios. In [1], techniques that make use of tight clock synchronization or GPS information to stop wormhole attacks are proposed. In [8], a distance bound on each link is acquired through a fast exchange of bits between sender and receiver. Other approaches toward thwarting the wormhole attack [9] rely on overhearing and statistical inference to detect the presence of a wormhole.

We propose TrueLink a practical countermeasure against the wormhole attack, presented here as an extension to the IEEE 802.11 MAC layer. TrueLink enables a node to verify the adjacency of an *apparent* neighbor, using a combination of timing and authentication. TrueLink is meant to be used together with a secure routing protocol. Authentication is an essential component of such protocols, and TrueLink can use any such mechanism for its own authentication needs.

TrueLink performs link verification between two nodes i and j in two phases: the **rendezvous** phase, and the **authentication** phase. In the **rendezvous** phase, i and j exchange $nonces$ α_j and β_i, where the subscript indicates the node that generated the nonce. This exchange proves the adjacency of the responding node through the use of strict timing constraints; only a direct neighbor is able to respond in time. In the **authentication** phase, i and j each sign and transmit the message (α_j, β_i), mutually authenticating themselves as the originator of their respective nonce. The timing constraints of the rendezvous phase makes TrueLink immune to capture-and-replay style wormhole attacks, and strictly limits the range of attacks based on bit-by-bit or "cut-through" forwarding. TrueLink combines many attractive features, which make it a good candidate for practical deployment:

A. Deployability with minimal requirements: TrueLink does not rely on precise clock synchronization, GPS coordinates, overhearing, or geometric or statistical methods.

B. Backwards compatibility with IEEE 802.11: TrueLink can be implemented using standard IEEE 802.11 hardware with a minor, backwards compatible, firmware update. We show how a $nonce$ can be included in each CTS frame without changing the frame format in Sec. VII. TrueLink-enabled terminals continue to interoperate with non-enhanced 802.11 hardware, albeit without TrueLink protection.

C. Compatible with most authentication methods: TrueLink

can be used equally well with asymmetric, symmetric, hash-based or other authentication mechanisms.

D. Widely applicable: TrueLink is independent of the routing protocol used, and improves the security of both proactive and reactive routing protocols.

We conduct analysis and simulations to show that a wormhole attack can potentially have a devastating effect on an unprotected network. We analyze a 100-node grid topology and we find that 4 attackers are enough to make 80% of the node pairs select paths that pass through a wormhole where, potentially, all payload packets could get dropped by the attackers. The potentially devastating effects of the wormhole attack are verified in our simulation results. TrueLink effectively protects the network against such attacks and the cost of protection with TrueLink is small.

Our work in perspective: TrueLink is a countermeasure against the classical wormhole attack. Several secure routing protocols have been proposed that safeguard wireless networks against other attacks, but that are vulnerable to the wormhole attack. In combination with a secure routing protocol, TrueLink can provide protection against a wide range of attacks possible on a wireless network. However, we acknowledge that in some scenarios, a strictly limited form of physical-layer wormhole attack may still be possible. We discuss the scope of this attack further in Sec. V.

In the next section, we discuss background and related work including secure routing protocols and previously proposed countermeasures to the wormhole attack. Sec. III describes TrueLink in detail. Sec. IV provides a security analysis, and V separately addresses the issue of physical layer wormholes. Sec. VI describes several ways in which TrueLink may be applied, depending on the scenario considered and the routing protocol in use. Sec. VII describes our proposed modification to the IEEE 802.11 standard. Performance evaluation results are provided in Sec. VIII, and Sec. IX concludes the paper.

II. Background and Related Work

In this section, we introduce the problem of secure routing. In particular, we discuss the wormhole attack and a number of previously proposed countermeasures that combat this attack.

Routing may be subject to a variety of attacks; to handle these, a wide variety of secure routing protocols and attack countermeasures have been proposed. The attacks can be divided into two major categories: attacks on the control plane, and attacks on payload traffic. Attacks on the control plane typically announce the presence of non-existent links or routes. Attacks on payload traffic, such as the Black Hole, Gray Hole, or Jellyfish attacks, are typically mounted by attackers that participate normally in routing control traffic, but drop, delay, modify or reorder some or all data packets that pass through them. Payload attacks can be particularly powerful in combination with a control-plane attack that is mounted to increase the amount of traffic routed through attacker nodes.

Many attacks can be countered using cryptography. This can help isolate all nodes that do not have the necessary credentials. Using cryptography is an attractive solution in many scenarios, as long as attackers are unable to *compromise* a node with the proper credentials. The wormhole attack, however, can be mounted without compromising the security of any node, and cannot be countered with cryptography alone.

A. The Wormhole Attack

In this paper, we study the *wormhole attack*; this is an attack on the routing control plane. The wormhole attack was introduced independently in [1], [2] and [3]. With this attack, one or more potentially colluding attackers record packets at one location, and replay them at another location, using either in-band (tunneling) or out-of-band communication to transfer the packets between these locations. This can give nodes that are in the neighborhood of the attackers the impression that links exist between them and other nodes that are in reality far outside of transmission range. The attacker **does not need to compromise any node**, or have any knowledge of the routing protocol in use. Since a wormhole attack involves recording and retransmitting packets verbatim, this attack can be mounted using only hardware introduced by the attacker.

By creating the illusion of a link, attackers may be able to manipulate nodes to send more traffic through them; this traffic may then be dropped, modified or recorded. In [1], Hu et al. describe how a wormhole attack can be used to disrupt routing in a wide variety of routing protocols, including DSR [5], AODV [4], DSDV [6] and OLSR [10]. In addition, most previously published secure routing protocols, such as [3], [11], [12], [7], are susceptible to this attack. For a survey of secure routing protocols, see [13].

In ODSBR [7], a reputation based scheme is used to rank all links in the network, and routes are selected based on link rankings. In this protocol, a wormhole would be treated as one or more links, which would be ranked with other links in the network. Awerbuch et al. claim that these links will be avoided if they exhibit byzantine behavior. However, they define the byzantine attack behavior in terms of packet drops only. This makes ODSBR vulnerable to the Jellyfish attack [14] in combination with a wormhole attack. In addition, a single wormhole could potentially fake a large number of links. With an average node degree of D, each wormhole could result in D^2 fake links, each of which would have to be individually detected by ODSBR before successful communication can be established.

B. Previous Wormhole Countermeasures

TrueLink improves the state of the art in wormhole countermeasures, being widely applicable and light-weight. In [1], two countermeasures are presented: geographical and temporal packet leashes. Leashes prevent wormhole attacks by letting the receiver of a packet determine if a packet has traveled further than the leash allows.

For geographical packet leashes, each packet is stamped, upon transmission, with the current geographical location of the sending node, and signed by the sender. The receiver of the packet compares the location of the sender to its own location, and is thus able to determine whether the sender is close enough to be a neighbor. Geographical packet leashes require accurate and verifiable location information.

With temporal leashes, all nodes have tightly synchronized clocks. The sender stamps the packet with the current time, and signs it for later authentication. The receiver compares the time in the packet with its local clock. If the difference exceeds some small value, determined by the maximum transmission range of the radio in use, the packet is discarded. Temporal packet leashes require extremely tight global clock synchronization, making it infeasible for many applications.

LiteWorp [9] relies on overhearing by selected nodes, called Guards, distributed throughout the network. The guards monitor local control traffic to detect wormhole attacks. LiteWorp assumes overhearing, omnidirectional antennas, and a static topology, making it infeasible for large classes of networks. Moreover, the reliance on overhearing may introduce a susceptibility to blackmail attacks through impersonation.

In [8], [15], the maximum distance between two neighboring nodes is bounded through a series of fast one-bit exchanges. These schemes use specialized hardware for accurate time measurement and fast switching between the send and receive modes. In [16], the Echo protocol is proposed, in which one or more nodes collaborate and use ultrasound to verify a node's claimed location. Directional antennas are used to prevent wormhole attacks in [17]. In [18], broadcast messages are protected by local broadcast keys, ensuring that remote nodes cannot decrypt packets that have traversed a wormhole. Here, the existence of special purpose guard nodes, with GPS and longer-range transmission capabilities are assumed. For a survey of location verification protocols, see [19].

IEEE 802.11 authentication. Existing techniques in IEEE 802.11 are not sufficient to protect against the wormhole attack. The IEEE 802.11 standard includes techniques (WEP, IEEE 802.11i) for node authentication. However, these techniques do not include strict timing requirements, making them highly vulnerable to the wormhole attack.

Byzantine Wormhole Attacks. Although the name sounds similar, the so-called "byzantine wormhole" attack belongs to an entirely different class of attacks than the classic wormhole attack. In a "byzantine wormhole" attack, the attacker has compromised one or more nodes, and uses the credentials from such nodes to create the appearance of, or announce the presence of, non-existing links. "Byzantine wormhole" attacks, or any other attacks involving compromised nodes, are best handled at the routing layer. Clearly, if nodes have been compromised, link verification cannot stop these from announcing non-existent links between themselves. Like TrueLink, previous work such as Packet Leashes [1], does not address the byzantine wormhole attack. In [20], forward error correction and multi-path routing are used to introduce path redundancy and improve delivery probability under adversarial conditions. On-Demand Secure Byzantine Routing (ODBSR) [21] has been shown to be highly resilient to variants of the byzantine wormhole attack. Other techniques, such as intrusion detection or tamper-proof hardware, have also been proposed to address the issue of compromised nodes.

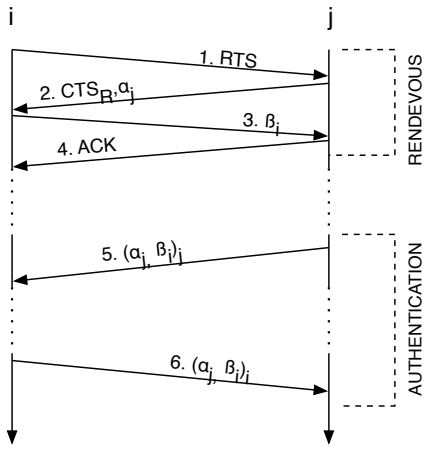

Fig. 1. Secure Link Verification. Node i is verifying that a direct link exists to node j. Non-essential transmissions excluded from figure.

III. TrueLink: Our Protocol

TrueLink verifies the adjacency of any neighbor, using a combination of timing and authentication. We present TrueLink as an extension to the IEEE 802.11 MAC layer. A TrueLink verification between two nodes i and j operates in two phases. In the **rendezvous** phase, i and j exchange $nonces$, randomly generated numbers. This phase is completed as a single RTS-CTS-DATA-ACK exchange. Here, timing constraints in the IEEE 802.11 standard make it extremely difficult for an attacker to successfully relay these frames (see Sec. V for more details). In the **authentication** phase, i and j each transmit a signed message (α_j, β_i), mutually authenticating themselves as the originator of their respective nonces.

Having established the adjacency of a neighbor, a routing protocol using TrueLink may use this information to constrain the reception and transmission of control and payload packets to links that have been verified to exist. Sec. VI discusses how to use TrueLink in combination with previously proposed routing protocols, and Sec. IV provides a security analysis of the TrueLink protocol.

Algorithm 1 (α, β) $Init()$

```
    sendRTS();
    # recvCTS() times out after t_timeout = SIFS
 3: α=recvCTS();
    if α != nil then
        β ← nonce();
 6:     # success = true if ACK received
        success=sendPacket([VerifyLink, β]);
        if success == true then
 9:         return (α, β);
        end if
    end if
12: return nil;
```

We now describe the operation of TrueLink in more detail. To verify the existence of a link, i initiates a link verification exchange, illustrated in Fig. 1. The exchange can be divided into a rendezvous phase, and an authentication phase.

Rendezvous Phase. The rendezvous phase is governed by strict timing constraints, making it extremely difficult for an

Algorithm 2 ($packet$, α) $HandleRTS()$

```
    if channelIsAvailable() then
        α ← nonce();
3:      sendCTS(α);
        # recvPacket times out after t_timeout = SIFS
        packet = recvPacket()
6:      if packet != nil then
            # returns packet together with α
            return (packet, α);
9:      end if
    end if
    return nil;
```

attacker to fake a link. When using the IEEE 802.11 MAC protocol as a basis, the rendezvous phase is implemented as a single RTS-CTS-DATA-ACK exchange. We describe the operation of the protocol for each of these frames. Pseudocode for the rendezvous operation is provided in Alg. 1 and 2.

1. RTS: Node i, the initiator of the link verification exchange, calls $Init()$. Node i sends an RTS to node j.

2. CTS, α_j: After receiving the RTS, node j calls $HandleRTS()$. A locally generated nonce α_j is included in the CTS, which is sent after a delay of one SIFS.

3. DATA, β_i: Having received the CTS, i generates nonce β_i. After a SIFS delay, the nonce is sent as the packet payload, together with a header identifying the packet as a rendezvous packet.

4. ACK: When node j receives the payload packet containing nonce β_i, the HandleRTS() function sends the ACK frame, after a SIFS delay. The received packet and the locally generated nonce α_j are handed to the upper layer for processing. When node i receives the ACK, the $Init()$ function at i returns the pair (α_j, β_i) to its caller.

Authentication. The authentication phase does not have any strong timing constraints. Nodes send the authentication messages as soon as they have finished computing the cryptographic signature. Node i sends the message $(\alpha_j, \beta_i)_i$ (signed by i) to j, thereby proving that it was the one sending the nonce β_i and that it received the correct α_j. This proves the existence of the link $i \leftrightarrow j$ to node j. Similarly, node j sends the message $(\alpha_j, \beta_i)_j$ to i. This proves the existence of the link to node i.

IV. Security Analysis

Previous sections provided a description and an intuitive argument as to how TrueLink protects against the wormhole attack. In this section, we more formally show that no MAC layer wormhole attack is possible. The next section separately addresses physical layer wormhole attacks. For ease of discussion, we will treat all colluding attacker nodes as a single entity. That is, attacker M consists of one or more nodes, spread throughout the network. M has the power to capture a frame from one legitimate node, say i, transfer the frame (in-band or out-of-band) to the vicinity of another legitimate node, say j, and replay it there.

In a classic wormhole attack, M will try to make it appear as if nodes i and j are neighbors, even though they are in reality out of range. We enumerate the ways in which M could attempt to do this. In a **forwarding** attack, M attempts to forward each frame, without modification, between i and j. This category also includes any delayed replay attacks. There are also two TrueLink-specific attacks. In a **masquerade** attack, M masquerades as j while responding to a link verification request initiated by i. M then initiates a link verification exchange with j while masquerading as i. Finally, in a **double masquerade** attack, M simultaneously initiates link verification exchanges with both nodes, while masquerading to each node as the other node, using the nonce received in the exchange with one node in its response in the other node. We will now show how TrueLink counters each of these attacks.

A. Forwarding attack. Let us first take a closer look at the forwarding attack, where M captures frames at one end, and replays them at the other. Without loss of generality, let us assume that i initiates the exchange. We will also assume that M does not know in advance the exact moment at which this exchange will be initiated.

M can decide to send an RTS to j after it hears the RTS from i. In this case, M does not have sufficient time to receive the CTS from j before $Init()$ on i times out. Alternatively, M could send an RTS to j *before* it hears the RTS from i. In this case, $recvCTS$ on j will time out while M is waiting for β_i from i. This is under the assumption that M does not know in advance the exact moment at which i will send its RTS. By default, functions $recvCTS$ and $recvPacket$ both time out after one SIFS interval (16 μs for IEEE 802.11b). By comparison, the shortest frames in the IEEE 802.11b standard, the CTS and ACK frames, take a minimum of 112 μs to transmit fully, not accounting for the physical preamble. Therefore, TrueLink is immune to store-and-forward attacks. Theoretically, there exists the possibility of a physical layer attack, where forwarding of a packet is done bit by bit, without first decoding the frame. We discuss this possibility in Sec V, and show how this attack can be countered as well. However, let us assume for now that this is outside of the attackers capabilities, and that 16 μs does not leave sufficient time for an attacker to synchronously rendezvous with j while i is waiting.

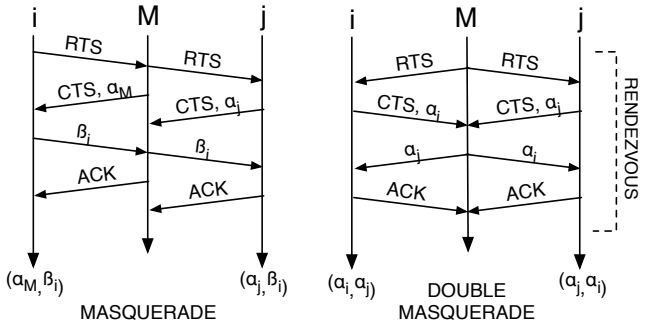

Fig. 2. Versions of the masquerade attack. Nonce pairs on i and j do not agree following the attack.

B. Masquerade attack. Fig. 2 illustrates the masquerade attack. Here, M instantly responds to the RTS from i by masquerading as j and generating its own nonce α_M. It then

initiates a second rendezvous with j, this time masquerading as i. At the end of the rendezvous phase, nodes i and j are using the nonce pairs (α_M, β_i), and (α_j, β_i) respectively. As these pairs are not equal, the verification exchange will be aborted in the authentication phase.

C. Double masquerade. In a double masquerade attack, M initiates link verification exchange between i and j. It accomplishes this by simultaneously masquerading as i when initiating the exchange with j, and as j when initiating the exchange with i. This method fails as well, as illustrated in Fig. 2 Neither i nor j actually initiated the exchange, and thus all nonces generated are α nonces. Even assuming α nonces are indistinguishable from β nonces, the resulting nonce pairs do not match, as shown in Fig. 2, and the exchange is aborted in the authentication phase.

A. Issues with Automatic Retransmission

A practical issue that must be addressed is the fact that most IEEE 802.11 cards will automatically retransmit DATA packets if a corresponding ACK is not successfully received. For TrueLink to remain secure, this feature must be temporarily turned off during the TrueLink rendezvous phase. The reason for this is that by sending the same nonce α or β in the CTS or DATA frame several times, a node potentially exposes its current value for α or β long before the actual TrueLink validation takes place. Doing so would introduce a security risk, where an attacker may capture the nonce the first time, and use this information to mount a wormhole attack.

V. Countering Physical Layer Wormholes

TrueLink is immune to conventional forwarding and tunneling. However, a physical layer repeater, which does not decode the signal, but merely replays the bits, or even the waveform received, may still be able to circumvent TrueLink. In this section, we address the theoretical limits on such an attack, and argue that with sufficiently small clock skew, the impact of a physical layer wormhole can be drastically reduced.

The IEEE 802.11 standard specifies that each frame in a RTS-CTS-DATA-ACK exchange be separated by a SIFS interval. This interval is either 10 or 16 microseconds, depending on the version of 802.11 used. The SIFS is inserted to provide transceivers sufficient time to switch between sending and receiving mode. If an expected frame is not received within a SIFS, the entire exchange is aborted. However, due to differences in hardware aspects, such as timer accuracy and switching times, and due to some variation in propagation delay stemming from the distance between sender and receiver, some allowance is made for variability in the duration of a SIFS. This slack, together with other weaknesses, can be exploited by a physical layer attacker to circumvent TrueLink.

We define **slack time** as an interval during which an attacker may have the opportunity to surreptitiously transfer a packet between wormhole end-points. An attacker has a number of techniques at his disposal for acquiring slack time. We will first outline the vulnerabilities, and then discuss in some detail how these are best addressed.

SIFS Slack Time. Nodes are allowed to start transmitting the next frame slightly before the full SIFS has passed, and the next frame can start arriving at the receiver a short time after a SIFS interval has passed without triggering a timeout.

Symbol Rate Attack. For similar reasons, some variation in the actual symbol rate of a sender is also allowed. Since the clock in the sender may run slightly faster or slower than the clock in the receiver, implementations sometimes accept up to a 10% variation in symbol rates [22].

Preamble Length. The purpose of a physical preamble is to allow the receiver to acquire the signal and achieve synchronization before frame transmission begins. As such, the receiver may sometimes not be able to decode the entire preamble of a frame. Instead, it will lock on to the signal some time during the preamble, with the end of the physical preamble indicated by the "start of frame" sequence.

The IEEE 802.11 standard postulates that transmission of a CTS, DATA or ACK frame must start within 1 SIFS of the completed reception of the preceding frame. This presents a wormhole vulnerability, as the attacker could conceivably transmit a shorter physical preamble than the standard requires, thereby creating a window of time in which the packet could be forwarded from a distant location.

A. Physical Layer Countermeasures

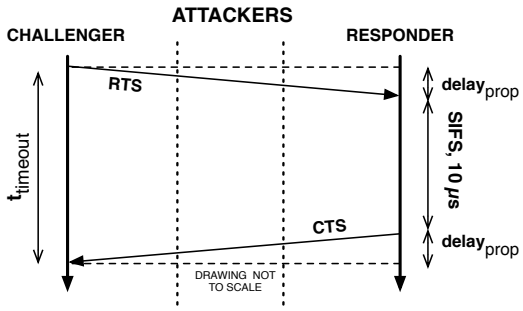

Fig. 3. The physical distance covered by a wormhole is limited by the timeout interval on the challenger side.

We address all three vulnerabilities with two enhancements to the standard operation of the protocol. First, we do not require clock synchronization. However, we do require clocks with a maximum skew of t_ϵ per bit-time, where a bit-time is the time required to transfer a bit of data at the basic transmission rate. Second, we redefine the timeout interval, after which a frame exchange is considered invalid.

The IEEE 802.11 standard incorporates slack time to allow for variation in clock speed, as well as in the distance between sender and receiver. With a maximum skew of t_ϵ per bit, a loose upper bound on the slack time available to an attacker can be expressed as

$$t_\epsilon \cdot s_{CTS} + s_{pre}/rate + r_{max}/c, \tag{1}$$

where r_{max} is the maximum allowed transmission range, s_{CTS} is the length of a CTS frame (including the preamble) in bits, s_{pre} is the length of a preamble in bits, and c is the

speed of light. Here, the first term represents the SIFS slack time, the second term represents the maximum time by which the preamble could be shortened, and the third term is the maximum propagation time.

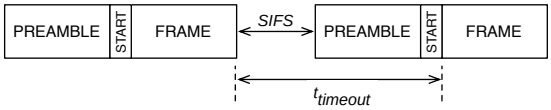

Fig. 4. The interval $t_{timeout}$ is re-defined as the time between the end of one transmission, and the beginning of the frame of the next. This counters the shortened preamble attack.

Using appropriate clocks [15], the first term of Eq. 1 can be made arbitrarily small. The second term allows the attacker to gain slack time by using a shorter physical preamble. This vulnerability arises from the manner in which the inter-frame timing constraints of the IEEE 802.11 protocol are defined. The time between the end of one frame and the beginning of the preamble of the next is defined to be one SIFS interval, as illustrated in Fig. 4.

We maintain identical timing constraints, but define the timeout interval to include the preamble, as indicated by $t_{timeout}$ in Fig. 4. With the redefined timing constraints, shortening the preamble does not gain the attacker any additional slack time.

We have now reduced Eq. 1 to r_{max}/c. This third term is unavoidable with TrueLink. Due to this, there exist circumstances under which a physical layer attacker (a sophisticated mode of attack in itself) may be able to create a wormhole despite TrueLink. Specifically, if two nodes are within the nominal transmission range, but still do not have a direct connection, due multipath or fading effects, these nodes may be vulnerable to attack. More formally, denoting the distance between nodes i, j as $d(i, j)$, the possibility of a successful wormhole attack exists when

$$d(n_c, wh_1) + d(wh_1, wh_2) + d(wh_2, n_r) < r_{max}, \quad (2)$$

where wh_1, wh_2 are the two endpoints of the wormhole, n_c, n_r are the challenger and responder nodes respectively, and where n_c and n_r do not have a direct connection. Recall that this attack requires sophisticated special purpose hardware and somewhat precise positioning of attacker nodes. In addition, even if the attack should succeed, the number of fake links created is limited by how many node pairs fulfill the criterion in Eq. 2.

VI. Link Verification Maintenance and Routing

Verifying the existence of a link is important in itself, but in a dynamic network, this knowledge needs to be maintained and properly updated as the topology changes. We present two methods of link verification maintenance with TrueLink.

Proactive Link Verification. With this method, a node periodically transmits beacon messages to find its current *apparent* neighbor set. It maintains knowledge about the *actual* neighbor set by periodically probing each neighbor with a link verification exchange. The proactive method is well suited to proactive routing protocols, or reactive protocols with local hello-messages. With these protocols, periodic beaconing maintains a consistent record of the apparent neighbors of a node, and it requires moderate extra effort to verify the links to each of these neighbors.

When a new neighbor is found, a link verification exchange is initiated before the routing layer is notified of the neighbor discovery. Moreover, after an expiry period t_{exp} has passed, each link is re-verified, to ensure that it still exists. The value of t_{exp} is flexible, but should not be smaller than the beaconing interval.

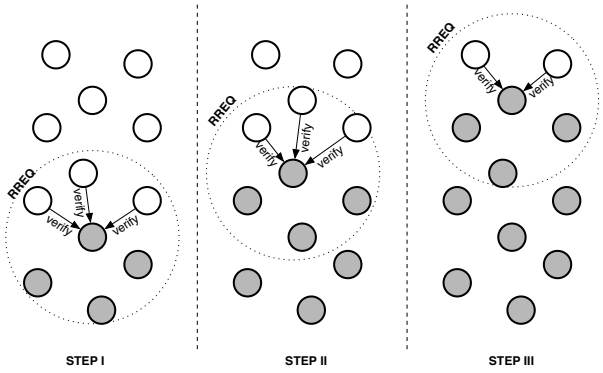

Fig. 5. Reactive link verification during a DSR Route Request phase. Receivers verify link existence after receiving a new route request over a previously unverified link.

Reactive link verification. Nodes using this method perform no link verification until the use of a link is needed. The reactive method is best suited to reactive routing protocols. As in the proactive case, a cache of recently verified neighbors is maintained, with an expiry interval t_{exp}. Figure 5 illustrates the reactive link-verification process for a broadcast route request. A node waits until it receives a broadcast packet across a previously unverified link. At this time, it initiates a link verification exchange with the sender of the packet. To avoid a potential "implosion" effect, where many receivers try to initiate an exchange at the same time, the receiver waits for a small randomly selected time interval before initiating the link verification exchange.

Overhead vs. Security Tradeoff. During the time interval t_{exp}, a node may believe that it has a link with a neighbor, when in fact the neighbor has moved away. This provides a window of opportunity for a wormhole attack faking the previously existing link. Thus, the value of t_{exp} provides a tradeoff between high overhead due to frequent verifications and attack vulnerability, due to potentially obsolete information. In our simulations, we use t_{exp} values between 10 and 120 seconds, with 30 seconds being the standard choice.

VII. MAC Layer Implementation

We propose to use 32 bits in each CTS frame to carry the $nonce$ required by TrueLink. We describe below how this can be done without changing the size of the CTS frame, while at the same time preserving backward compatibility.

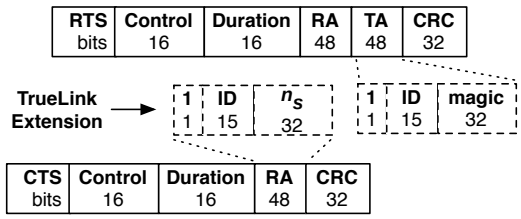

Fig. 6. The format of IEEE 802.11 RTS/CTS frames, and the modifications made to accomodate TrueLink.

Format of IEEE 802.11 RTS/CTS frames. Fig. 6 specifies the format of the IEEE 802.11 RTS/CTS frames. For the remainder of this discussion, we will denote a field X of a frame Y as Y:X. The RA and TA fields contain the MAC address of the receiver and transmitter, respectively. In the standard implementation, the value of CTS:RA is copied from the RTS:TA field of the immediately preceding RTS frame. The purpose of RTS:RA is to identify the intended recipient. This is necessary for the correct operation of the protocol.

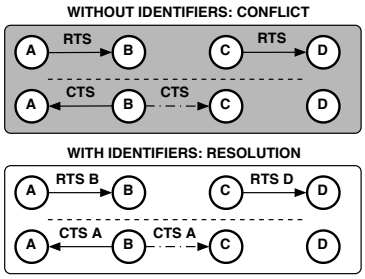

Fig. 7. The purpose of the CTS:RA field is to disambiguate between transmitters. This does not require 48-bit precision.

Revisiting CTS:RA. The main purpose of CTS:RA is to disambiguate between transmitters that happened to send their RTS frames simultaneously. For example, in Fig. 7, two nodes (A, C) are transmitting send requests simultaneously. Without CTS:RA to disambiguate, sender C may interpret the CTS B $\rightarrow$ A, as a response from D.

However, we claim that CTS:RA does not require 6 byte precision for successful disambiguation. We reason as follows: First, it is somewhat unlikely that the situation described in Fig. 7 would occur in the first place. Second, using an n-bit random number for disambiguation instead of a 6-byte address, we can disambiguate between the two simultaneous RTS frames with a probability $1 - 1/2^n$.

Thus, some of the bits in the CTS:RA and RTS:TA fields can be freed up for other purposes. For the purpose of this discussion, we will set the most significant bit, MSB=1, to indicate a locally administered MAC address (per the Ethernet address standard), use 15 bits for disambiguation purposes, and the remaining 32 bits of the CTS:RA field for storing a $nonce$, as shown in Fig. 6.

Stations implementing TrueLink need to change the contents of their outgoing RTS:TA fields. Specifically, MSB=1 to indicate a locally administered MAC address, the 15 following bits contain a randomly generated number, and the remaining 32 bits contain a special *magic* bit sequence recognized by all TrueLink implementations.

A. Coexistence with IEEE 802.11

The modification to the CTS frame that we have proposed is compatible with the IEEE 802.11 MAC standard. Specifically, we require sender and receiver to both implement our modifications. However, the modifications will not interfere with the operation of devices not implementing our modification.

The standard specifies the content of the CTS:RA header to be the MAC address of the receiver. However, only the intended recipient of the frame needs to recognize that the packet is intended for it. Other nodes use only the duration field of RTS/CTS frames for "virtual carrier sensing". Here, nodes wait for the duration specified before attempting a new transmission. It follows that nodes implementing TrueLink can peacefully coexist with standard IEEE 802.11 terminals.

TrueLink nodes can identify other TrueLink nodes without incurring any additional messaging overhead.

Recognizing TrueLink sender stations. The receiver station quickly identifies TrueLink senders by inspecting RTS:TA. If the sender implements TrueLink, MSB=1 and the lower 32-bits will contain the *magic* bit sequence.

Recognizing TrueLink receiver stations. As specified earlier, a receiver implementing our extension will copy the 16 most significant bits from RTS:TA to its CTS:RA, and set the remaining 4 bytes of CTS:RA to a nonce generated locally. By contrast, an IEEE standard implementation will respond with CTS:RA = RTS:TA. When the sender receives a CTS, it inspects CTS:RA. If the lower order 32 bits are still set to the *magic* bit sequence, the sender can determine that other receiver station does not implement TrueLink, and act accordingly. Note that TrueLink cannot protect communication with standard IEEE 802.11 devices against wormhole attacks.

VIII. Performance Evaluation

In this section, we evaluate various aspects of the performance of TrueLink. Results from an actual implementation of TrueLink would have been a valuable addition to this section. Unfortunately, the firmware for readily available 802.11 hardware is proprietary, and we have not been able to acquire the necessary source code. However, while an implementation would have provided strong evidence as to the feasibility of TrueLink, the performance and security of TrueLink are better evaluated through other means.

A. On the strength of the Wormhole Attack

We discuss the strength of a wormhole attack in terms of the number of node pairs it covers. By cover, we mean that during a wormhole attack, the shortest perceived path between the two nodes will always go through the attackers.

Let us assume we have a network containing nodes v_a and v_b that are trying to establish a connection, and attackers $m_0 \ldots m_{max}$. Further, let us denote the distance between any two nodes $i,\ j$ as $d(i, j)$. For these calculations, we will assume that wormholes have been established between all pairs of attackers.

Under a wormhole attack, nodes v_a and v_b will pick a working path, for any pair of attackers m_i and m_j,

$$dist(v_a, m_i) + dist(v_b, m_j) - 1 > d(v_a, v_b). \quad (3)$$

To see why, consider that $dist(m_i, m_j) = -1$ under a wormhole attack. How many node pairs are d hops apart? For ease of analysis, we will assume a grid topology, and use the manhattan distance. In the one-dimensional case, with a string of length W, we have $(W - d)$ possible pairs that are a distance d apart.

In the two-dimensional case, with a grid of size (W, H), we divide the distance d into a horizontal component x, and a vertical component $d - x$. For each x there are $2(W - x)$ possible locations for the first node, and there are another $2(H-(d-x))$ possible locations for the second node. Dividing by 2 to get the number of unordered pairs, we obtain the number of pairs that are d hops distant in an (W, H) grid as

$$\frac{4}{2}\sum_{x=1}^{d}(W - x)(H - d + x)$$

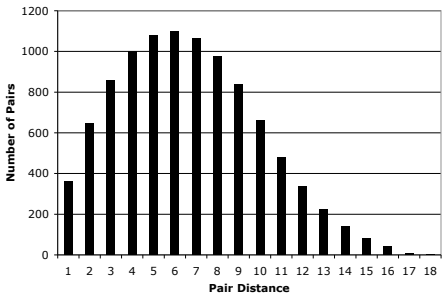

Fig. 8. Distribution of distances between all pairs in a 10x10 grid.

Hops (x)	p(x)
1	1/6
2	1/3
3	1/3
4	1/6

TABLE I

PROBABILITY OF A GIVEN NODE BEING X HOPS AWAY FROM AN ATTACKER. 10X10 GRID, 4 ATTACKERS.

Hops (x)	p(x)	cumulative
1	1/18	1/18
2	1/9	3/18
3	2/9	7/18
4	2/9	11/18
5	2/9	15/18
6	1/9	17/18
7	1/18	1

TABLE II

PROBABILITY THAT FOR A GIVEN PAIR OF NODES, THE SUM OF THEIR RESPECTIVE DISTANCES TO AN ATTACKER EQUALS X, IN A 10X10 GRID NETWORK WITH 4 ATTACKERS.

Network Size	100 nodes
Network Dimensions	1400x1400 m
Radio Range	250 m
Channel Capacity	1 mbit/s
Packet Size	512 b
Offered Load (total)	200 pkts/sec
Number of Flows	1-50
TrueLink Cache Timeout	5-120 s
Traffic Type	CBR / UDP
Routing Protocol	AODV
Max Rate of Mobility	1-20 m/s
Mobility Model	Random Waypoint
Warmup Period	3000 s

TABLE III

PARAMETERS USED IN SIMULATION EXPERIMENTS.

Figure 8 shows the distance between all pairs in a 10x10 grid, using the equation above. With 4 well-placed attackers, the probability of a given node being x hops away from an attacker, $p(x)$, is shown in Tab. I. Given that pairs are independently sampled, we can calculate the probability of Eq. 3 being true for all d < 8. The results are shown in Tab. II.

We can now calculate the estimated number of pairs failing to find a route due to the wormhole attack, using Eq. 3. Solving numerically for a 10x10 grid with 4 attackers, we find that 79.5% of all pairs fail to find a path that does not pass through a wormhole. Clearly, the wormhole attack is a significant concern. Most pairs fail to find a working path, and all pairs with a distance > 7 will invariably fail to find a good path in this scenario.

B. Simulation Results

We now provide simulation results to substantiate the analysis above. For our simulation work, we used the ns-2.29 simulation environment. Tab. 3 shows the parameters used in our experiments. For these experiments, we used the existing AODV implementation in ns-2.29, extended to support TrueLink. The code for the TrueLink extension, as well as the code for simulating a wormhole attack, is available at [23]. UDP/CBR flows were used in all the presented simulation work. TCP/FTP flows yield similar results. However, the adaptive nature of such flows introduce additional factors, such as shorter flows receiving better throughput, that make the results difficult to interpret. Reactive TrueLink link verification (see Sec. VI) was employed for every received AODV packet. Each verification was simulated as a rendezvous exchange followed by two separate signed messages, as described in Sec. III. Received AODV packets were buffered until the TrueLink exchange had been completed, and then processed as usual. Any key exchange protocol or other cryptographic messaging in addition to the signatures used was not modeled. We assume the use of elliptic curve cryptography for signatures, with 160-bit keys (equivalent crypto strength to 1024-bit RSA keys [24], or 80-bit keys using symmetric techniques).

C. Link Verification Overhead

Protecting a routing protocol with TrueLink incurs some amount of overhead. As described in Sec. III, each link

verification exchange requires one rendezvous packet and two authentication packets. The extra packets transmitted may result in additional interference, packet drops and consequently route breakage. This may in turn set off additional route requests and increase AODV control traffic. We are interested in determining the additional overhead incurred by TrueLink, both in terms of link verification exchanges, and in terms of these secondary effects on AODV from the delays incurred and the extra load on the network. For these results, no wormhole attacks were simulated.

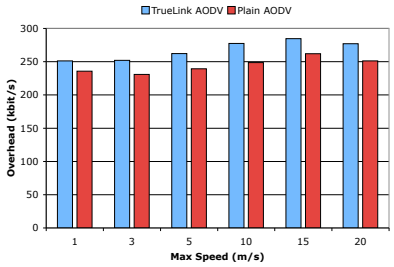

Fig. 9. Overhead vs. Maximum Node Speed. TrueLink overhead is relatively small ($< 10\%$), and largely independent of mobility.

TrueLink incurs low overhead. Figure 9 shows the overhead incurred for varying rates of mobility, for Plain AODV and TrueLink AODV. As mobility increases, TrueLink AODV experiences a slightly higher increase in overhead as compared to Plain AODV. This is likely due to a somewhat increased number of neighbors encountered for each node participating in a route lookup. We also evaluated TrueLink performance with respect to the number of flows, and the t_{exp} timeout value. In short, TrueLink was not significantly affected by flow count, and TrueLink overhead exhibited a significant linear drop as the expiry time was increased. Due to space limitations, we cannot include the result plots.

D. Performance During Attack

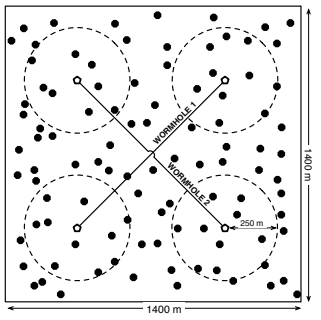

Fig. 10. Typical wormhole attack scenario. Attacks were simulated with 0, 1 or 2 wormholes in the shown configuration.

Here we demonstrate the benefit of TrueLink when the network is besieged by a wormhole attack. The purpose of these experiments is to further evaluate the strength of the wormhole attack, and to verify that the overhead incurred by TrueLink results in a significant improvement. As we shall see, TrueLink improves average throughput in many scenarios, and drastically improves overall network reliability.

We implemented the wormhole attack in ns-2.29 through modifications to the MAC and LL components. In the link-layer component, packets are forwarded out-of-band through to the other end of the wormhole. To simulate a black-hole attack, all payload packets were dropped by the attackers. The MAC layer was modified to allow the reception of packets with a destination MAC address equal to any neighbor of the attacker at the other end of the wormhole and to respond appropriately to RTS frames addressed to such nodes. All other simulation parameters were kept the same as in previous experiments. We chose to simulate 3 different attack scenarios, denoted (0), for no wormholes, (1), for a single wormhole, and (2) for two wormholes in a cross configuration. Figure 10 illustrates the attack scenarios. Wormhole nodes were sufficiently separated that any given frame would be received by at most one wormhole node. These scenarios simulate up to 2 wormholes with up to 4 attackers. Note that 4 attackers could potentially form up to 6 wormholes, by establishing one wormhole for every pair of attackers. Such an attack may even further degrade the performance of an unprotected network.

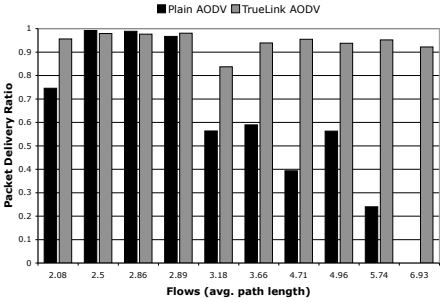

Fig. 11. Packet Delivery Ratio vs. Average Path Length of each flow. Flows with longer paths tend to be susceptible to the wormhole attack.

Distant source-destination pairs suffer with Plain AODV. Plain AODV consistently loses all or almost all packets of one or more flows. Fig. 11 shows a detailed, per-flow graph for a 2-wormhole scenario with an 80 kbps/s total offered load across 10 flows. Packet delivery ratios for the flows using longer paths are highly affected by the wormhole attack. However, when TrueLink is employed, the effect of the wormhole attack is removed entirely. Note that due to mobility, several flows are only partially affected by the wormhole attack (as evidenced by the difference between Plain AODV and TrueLink AODV).

Average throughput increases with TrueLink. In Fig. 12, the scenario consisted of 100 nodes and 10 flows. Mobility speed was 1 m/s. Each group of bars represents a set of experiments, at a given offered load. Each bar represents a single simulation experiment, each with a given routing protocol and attack scenario. The bracket overlaid on top of the bar shows the minimum and maximum throughput achieved by any given flow in that specific experiment.

For the lowest offered load, 40 kbps, we note that the entire load (10 * 4 kbps) is delivered by TrueLink AODV in all cases,

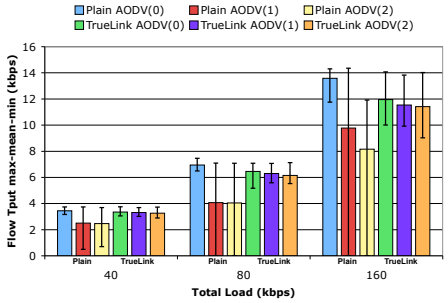

Fig. 12. Flow Throughput vs. Send Rate. Brackets show max and min flow throughput, revealing the weakness of unprotected AODV.

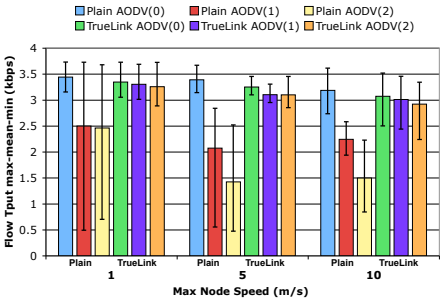

Fig. 13. Packet Delivery Ratio vs. Node Mobility. Wormhole attack has less severe effects with high node mobility.

whereas Plain AODV did so only for the 0-attacker case. For the larger loads, neither protocol achieves a 100% delivery ratio, with delivery ratio decreasing as load increases. This is simply an effect of the 1 mbps channel being saturated, keeping in mind that most connections span more than a single hop. As shown earlier in this section, TrueLink AODV consistently achieves marginally lower throughput than Plain AODV for the 0-attacker case. In cases with 1 or 2 attackers, however, AODV with TrueLink is clearly the better choice, with higher throughput in all but one scenario.

Mobility distributes attack consequences over more pairs. Figure 13 contrasts packet delivery ratio and node mobility. Even though average throughput does vary to some degree, a more interesting result is to be found in the max-min brackets. As node mobility increases, the effect of the wormhole attack is spread more evenly across the nodes in the network. At a maximum speed of 10 m/s, throughput variability is similar for Plain AODV and TrueLink AODV, however, TrueLink AODV achieves twice the throughput.

IX. Conclusion

We propose TrueLink, a countermeasure against the wormhole attack, which can be implemented *now* on IEEE 802.11 networks with a minor firmware update to existing commercial hardware. Wormhole protection is essential to a secure routing protocol, and we believe TrueLink is good candidate for this. Specifically, TrueLink has the following attractive properties.

Ready to be deployed. It has minimal support requirements. TrueLink does not rely on precise clock synchronization, GPS coordinates, overhearing, or statistical methods.

Backwards compatible with IEEE 802.11. We design TrueLink to be interoperable with IEEE 802.11. Thus, it can be used with the current commercial hardware with only a minor, backwards compatible, firmware modification.

Applicable to most routing protocols. TrueLink can be integrated into virtually any routing protocol.

The wormhole attack is an important attack that has yet to see a general purpose solution. With TrueLink, we hope to remedy this situation. TrueLink is well suited for integration in most existing secure routing protocols, and combined with the state of the art in secure routing, promises to bring secure wireless networks one step closer to reality.

References

[1] Y. Hu, A. Perrig, and D. Johnson, "Packet leashes: A defense against wormhole attacks in wireless networks.," in *INFOCOM*, 2003.

[2] P. Papadimitratos and Z. Haas, "Secure routing for mobile ad hoc networks," in *Proc. SCS Communication Networks and Distributed Systems Modeling and Simulation*, 2002.

[3] K. Sanzgiri, B. Dahill, B. Levine, C Shields, and E. Belding-Royer, "A secure routing protocol for ad hoc networks," in *ICNP*, 2002, pp. 78–89.

[4] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Proc. IEEE Workshop on Mobile Computing Systems*, 1999.

[5] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad-hoc wireless networks," in *Mobile Computing*. Kluwer Academic, 1996.

[6] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance vector routing (dsdv) for mobile computeres," in *Proc. ACM SIGCOMM*, Sep. 1994.

[7] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens, "An on-demand secure routing protocol resilient to byzantine failures," in *WiSE '02*, 2002.

[8] Srdjan Capkun, Levente Buttyn, and Jean-Pierre Hubaux, "Sector: secure tracking of node encounters in multi-hop wireless networks," 2003.

[9] Issa Khalil, Saurabh Bagchi, and Ness B. Shroff, "Liteworp: A lightweight countermeasure for the wormhole attack in multihop wireless networks.," in *DSN*, 2005, pp. 612–621.

[10] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized link state routing protocol for ad hoc networks," in *IEEE INMIC*, 2001.

[11] Y. Hu, A. Perrig, and D Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," in *ACM MobiCom*, 2002.

[12] Y. Hu, D. Johnson, and A. Perrig, "Sead: Secure efficient distance vector routing in mobile wireless ad hoc networks," in *WMCSA '02*, 2002.

[13] Yih-Chun Hu and Adrian Perrig, "A survey of secure wireless ad hoc routing.," *IEEE Security & Privacy, special issue on Making Wireless Work*, May/June 2004.

[14] Imad Aad, Jean-Pierre Hubaux, and Edward W. Knightly, "Denial of service resilience in ad hoc networks," in *MOBICOM*, 2004.

[15] Stefan Brands and David Chaum, "Distance-bounding protocols," in *Advances in Cryptology – EUROCRYPT ' 93*.

[16] Naveen Sastry, Umesh Shankar, and David Wagner, "Secure verification of location claims.," in *Workshop on Wireless Security*, 2003.

[17] Lingxuan Hu and David Evans, "Using directional antennas to prevent wormhole attacks.," in *NDSS*, 2004.

[18] Radha Poovendran and Loukas Lazos, "A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks," *ACM Journal on Wireless Networks (WINET)*, 2005.

[19] I. Broustis, M. Faloutsos, and S. Krishnamurthy, "How does topology affect security in wireless network?," in *eSCo-Wi*, 2006.

[20] Panagiotis Papadimitratos and Zygmunt J. Haas, "Secure data transmission in mobile ad hoc networks," in *WiSe '03*, New York, NY, USA, 2003, pp. 41–50, ACM Press.

[21] B. Awerbuch, R. Curtmola, D. Holmer, H. Rubens, and C. Nita-Rotaru, "On the survivability of routing protocols in ad hoc wireless networks," in *SecureComm*, 2005.

[22] Mart Molle, "Private correspondence," 2005.

[23] J. Eriksson, "Source code for truelink and wormhole attack in ns-2," http;//www.cs.ucr.edu/ jeriksson/, 2005.

[24] A.J. Menezes, *Elliptic curve public key cryptosystem*, Kluwer Academic Publications, 1993.