Wormhole Detection in Wireless Ad Hoc Networks

Yih-Chun Hu
Carnegie Mellon University
yihchun@cs.cmu.edu

Adrian Perrig
UC Berkeley
perrig@cs.berkeley.edu

David B. Johnson
Rice University
dbj@cs.rice.edu

Rice University Department of Computer Science
Technical Report TR01-384
December 17, 2001
Revised: June 15, 2002

Abstract

As mobile ad hoc network applications are deployed, security emerges as a central requirement. In this paper, we introduce the *wormhole attack*, a severe attack against ad hoc routing protocols that is particularly challenging to defend against. We show how an attacker can use the wormhole attack to cripple a range of ad hoc network routing protocols. In the wormhole attack, an attacker records packets (or bits) at one location in the network, tunnels them to another location, and retransmits them there into the network. Most existing ad hoc network routing protocols, without some mechanism to defend them against the wormhole attack, would be unable to find routes longer than one or two hops, severely disrupting communication.

In this paper, we present the design of two protocols capable of detecting a wormhole attack at the receiver: the Slot Authenticated MAC protocol and the TIK protocol. Both protocols rely on tight time synchronization, which can be readily available through off-the-shelf hardware. These two protocols make different tradeoffs: the Slot Authenticated MAC is a simple, very resource-efficient approach based on a TDMA MAC, whereas TIK is a practical, novel approach that has somewhat higher network overhead and resource requirements, but features significantly reduced latency. The computational requirements of TIK are well within the range of existing handhelds computers. For example, in an 11Mbps wireless LAN, a Compaq iPaq 3870 PocketPC is capable of handling the maximum rate of cryptographic operations associated with TIK, using just 3% of its built-in memory and 18% of its CPU time.

1 Introduction

The promise of mobile ad hoc networks to solve challenging real-world problems continues to attract attention from industrial and academic research projects. Applications are emerging and widespread adoption is on the horizon. Most previous ad hoc network research has focused on problems such as routing and communication, assuming a trusted environment. However, many applications run in untrusted environments and require secure communication and routing. Applications that may require secure communications include emergency response operations, military or police networks, and safety-critical business operations such as oil drilling platforms or mining operations. For example, in emergency response operations such as after a natural disaster like a flood, tornado, hurricane, or earthquake, ad hoc networks could be relied upon for real-time safety feedback. Regular communication networks may be damaged, so emergency rescue teams might rely upon ad hoc networks for communication. To fend off malicious attackers in these emergency situations, these safety-critical applications require secure communication.

Ad hoc networks generally use a wireless radio communication channel. The main advantage of such networks is low cost of deployment and maintenance, since the nodes and wireless hardware are inexpensive and readily available, and since the network is automatically self-configuring and self-maintaining. However, wireless networks are vulnerable to several attacks. An attacker can *spoof*, or impersonate, another user on the network. *Replay* attacks are

also a concern, in which an attacker records a message and replays it again later. A particularly challenging attack to defend against is a *wormhole* attack, in which an attacker records a packet, or individual bits from a packet, at one location in the network, tunnels the data to another location, and replays it there. (We describe the wormhole attack in more detail in Section 2.1.) An attacker can use these attacks to gain unauthorized access, disrupt routing, or perform a denial-of-service attack (DoS).

In this paper, we present two protocols that defend against wormhole attacks: the Slot Authenticated MAC protocol and the TESLA with Instant Key disclosure protocol (TIK). Both protocols rely on very accurate time synchronization and utilize the TESLA authentication protocol. TIK is a layer on top of existing MAC protocols that provides secure authentication and freshness, specifically for the needs of security in ad hoc networks. The Slot Authenticated MAC protocol uses TDMA MACs to provide link-layer authentication and wormhole detection for nodes with fewer resources.

Section 2 discusses the problem solved by TIK and describes the attacker model. In Section 3, we present our assumptions. Section 4 reviews the TESLA authentication protocol, on which our protocols are based, and Section 5 introduces the Slot Authenticated MAC protocol and a "straw man" version of TIK. In Section 6, we present a hash tree extension to TIK for efficient key authentication, which we use in Section 7 to make a practical version of the TIK protocol; Section 7 also discusses the integration of TIK with several types of MAC layer protocols. Section 8 analyzes the efficiency of TIK in real-world circumstances, Section 9 presents a general approach for detecting wormholes, Section 10 discusses related work, and Section 11 presents our conclusions.

2 Problem Statement

Wireless communication faces several security risks. An attacker can easily *inject* bogus packets, impersonating another sender. We refer to this attack as a *spoofing* attack. An attacker can also easily *eavesdrop* on communication, record packets, and *replay* the (potentially altered) packets. A powerful attack against many ad hoc network routing protocols is the *wormhole attack*. An attacker can use these attacks to gain unauthorized access, compromise systems, or perform denial-of-service (DoS) attacks. We first describe the wormhole attack, and then discuss possible countermeasures and foreshadow our solutions.

2.1 The Wormhole Attack

In a *wormhole attack*, an attacker receives packets at one point in the network, "tunnels" them to another point in the network, and then replays them into the network from that point. For tunneled distances longer than the normal wireless transmission range of a single hop, it is simple for the attacker to make the tunneled packet arrive sooner than other packets transmitted over a normal multihop route, for example by use of a single long-range directional wireless link or through a direct wired link to a colluding attacker. It is also possible for the attacker to forward each bit over the wormhole directly, without waiting for an entire packet to be received before beginning to tunnel the bits of the packet, in order to minimize delay introduced by the wormhole.

The wormhole attack is a particularly dangerous attack against many ad hoc network routing protocols in which the nodes that hear a single-hop transmission of a packet consider themselves to be in range of (ant thus neighbors of) the sender.

For example, when used against an on-demand routing protocol such as DSR [20] or AODV [32], a powerful application of the wormhole attack can be mounted by tunneling each ROUTE REQUEST packet directly to the destination target node of the REQUEST. When the destination node's neighbors hear this REQUEST packet, they will follow normal routing protocol processing to rebroadcast that copy of the REQUEST and then discard without processing all other received ROUTE REQUEST packets originating from this same Route Discovery. This attack thus prevents any routes other than through the wormhole from being discovered, and if the attacker is near the initiator of the Route Discovery, this attack can even prevent routes more than two hops long from being discovered.

The neighbor discovery functionality of periodic (proactive) routing protocols such as DSDV [31], OLSR [39], and TBRPF [5] rely heavily on the reception of broadcast packets as a mechanism for neighbor detection, and are also

extremely vulnerable to this attack. For example, OLSR and TBRPF use HELLO packets for neighbor detection, so if an attacker tunnels to B all HELLO packets transmitted by A, and tunnels to A all HELLO packets transmitted by B, then A and B will believe that they are neighbors, which would cause the routing protocol to fail to find routes when they are not actually neighbors.

If each DSDV advertisement sent by node A were tunneled to node B, and vice versa, then A and B would believe that they were neighbors. If they were not within wireless transmission range, they would be unable to communicate. Furthermore, if the best existing route from A to B were at least $2n + 2$ hops long, then any node within n hops of A would be unable to communicate with B, and any node within n hops of B would be unable to communicate with A. Otherwise, suppose C were within n hops of A, but had a valid route to B. Since A advertises a metric 1 route to B, C would hear a metric $n + 1$ route to B. C will take that route if it is not within $n + 1$ hops of B, in which case there would be a n-hop path from A to C, and a $n + 1$-hop path from C to B, contradicting the premise that the best real path from A to B is at least $2n + 2$ hops long.

2.2 Detecting and Preventing Wormholes

The general approach we take in preventing wormhole attacks is to measure the packet propagation delay (time from when a legitimate node sent the packet to when the receiver receives the packet) and to discard packets that traveled too far. This approach requires tight time synchronization, but prevents the wormhole attack without requiring secure hardware.

An alternative approach to preventing wormhole attacks is to use a secret method for modulating bits over the air; once a node is compromised, however, this approach is likely to fail unless the radio is kept inside tamper-resistant hardware. Another approach, known as RF watermarking, authenticates a transmission without decoding the data by modulating the RF waveform in a way known to authorized nodes [12]. RF watermarking relies on keeping secret the knowledge of which RF waveform parameters are being modulated; furthermore, if that waveform is exactly captured at the receiving end of the wormhole and exactly replicated at the transmitting end of the wormhole, the signal level of the resulting watermark is independent of the distance it was tunneled. As a result, the watermark may still be intact, even though the packet was made to travel beyond the valid wireless transmission range. Though intrusion detection could be used in some cases to detect a wormhole attacker, it is generally difficult to isolate the attacker in a software-only approach, since the packets sent by the wormhole are identical to the packets sent by legitimate nodes.

The key insight to our mechanisms is that *authentication* coupled with tight time synchronization provide a mechanism for *freshness* and wormhole detection. When a receiver authenticates the sender of a packet, the receiver is certain that the packet originated from the sender and was not altered by an attacker. Most simply, if a timestamp is placed by the transmitting node in the authenticated portion of each packet, the elapsed time from transmission to receipt can be computed to within the accuracy of time synchronization, thus providing freshness and enabling wormhole detection.

Most ad hoc network communication is *unicast* (point-to-point) communication, but most routing protocols and some applications also rely on *broadcast* communication. Therefore, methods used to secure ad hoc network communication should be applicable for both unicast and broadcast communication. Unfortunately, securing broadcast communication is a much harder problem than securing unicast communication [34]. We have designed the Slot Authenticated MAC protocol and the TIK protocol each to secure both unicast and broadcast communication.

Message authentication codes are a standard approach for authentication: the sender S and receiver R must share a secret key K, which they use in conjunction with a message authentication code function (for example HMAC [4]) to authenticate messages they exchange. To send a message M to R, S sends:

$$S \rightarrow R: \quad \langle M, \text{HMAC}_K(M) \rangle$$

where the notation $\text{HMAC}_K(M)$ represents the HMAC message authentication code computed over message M with key K. The packet sent from S to R contains both the intended message M and $\text{HMAC}_K(M)$. When R receives this message, it can verify the authenticity of the message by comparing the received HMAC value to the HMAC that it computes for itself over the received message with the secret key K it shares with the sender S.

However, using message authentication codes in the standard way has two major drawbacks. First, in a network with n nodes, we would need to set up $\frac{n(n-1)}{2}$ keys, one for each pair of nodes. Key setup is an expensive opera-

tion, which makes this approach impractical in large networks. Second, this approach cannot efficiently authenticate broadcast packets. To secure a broadcast packet, the sender would need to add to the packet a separate message authentication code for each receiver, making the packet extremely large (and likely exceeding the network's maximum packet size). The need to include separate message authentication codes in the packet could be avoided by having multiple receivers share the same key, but this might allow a subset of colluding receivers to impersonate the sender [8].

Instead, attaching a *digital signature* to each packet could be used to solve the above two problems: each node needs to have only one public-private key pair, and each node needs to know only the public key of every other node; thus, only n public keys need to be distributed. Furthermore, a digital signature provides authentication for broadcast packets in the same way as for unicast packets. (A digital signature also provides *non-repudiation*, which is a stronger property than authentication because the receiver can not only convince herself of the origin of the packet, but also a third party. Fortunately, the majority of applications only require authentication, which we can achieve with more efficient methods.)

However, digital signatures have some drawbacks. First, digital signatures are usually based on computationally expensive *asymmetric* cryptography. For example, the popular 1024-bit RSA digital signature algorithm [40], roughly equivalent to use of a 72-bit key in a symmetric encryption algorithm [25], requires about 10 milliseconds on a 800 MHz Pentium III processor for signature generation. Signature verification is more efficient, but still requires about 0.5 milliseconds. Adding a digital signature to each packet is computationally expensive for the verifier (receiver), but overwhelmingly expensive for the signer (sender). On less powerful CPUs, each digital signature generation and verification takes on the order of seconds [7]. Even with recent advances in digital signatures [11, 26, 37], digital signatures are impractical for authenticating communication on a packet-by-packet basis over a moderate speed link (capable of carrying a reasonable number of packets per second). In addition, for security, a digital signature must be larger (more bits) than a mesage authentication code, and thus this approach consumes more network bandwidth.

Our Slot Authenticated MAC and TIK (TESLA with Instant Key disclosure) protocols are designed to support the specific security requirements of ad hoc networks. These protocols each provide efficient authentication, freshness, and wormhole detection/prevention for unicast *and* broadcast messages at the MAC layer, and are based on efficient *symmetric* cryptographic primitives (a message authentication code is a symmetric cryptographic primitive). Both of these protocols require accurate time synchronization between all communicating parties; each also only requires each communicating node to know just one public value for each sender node, enabling scalable key distribution.

3 Assumptions and Notation

The acronym "MAC" may stand for Medium Access Control protocol and Message Authentication Code; to avoid confusion we use "MAC" to refer to the network Medium Access Control protocol at the link layer, and we use "HMAC" to refer to a message authentication code used for authentication (HMAC is a particular instance of a message authentication code [4]).

For reasons such as differences in wireless interference, transmit power, or antenna operation, links between nodes in a wireless network may at times work in only one direction; such a *unidirectional* wireless link between between two nodes A and B might allow A to send a packet to B but not for B to send a packet to A. In many cases, however, wireless links operate as *bidirectional* links. A MAC protocol generally is designed for operation over unidirectional links or only for bidirectional links; the introduction of either the Slot Authenticated MAC or TIK do not affect the capability of the MAC protocol to send over unidirectional links.

Security attacks on the wireless network's physical layer are beyond the scope of this paper. Spread spectrum has been studied as a mechanism for securing the physical layer against jamming [36]. Denial-of-Service (DoS) attacks against MAC layer protocols are also beyond the scope of the paper; MAC layer protocols that do not employ some form of carrier sense, such as pure ALOHA and Slotted ALOHA [1], are less vulnerable to DoS attacks, though they tend to use the channel less efficiently.

We assume that the wireless network may drop, corrupt, duplicate, or reorder packets. We also assume that the MAC layer contains some level of redundancy to detect randomly corrupted packets; however, this mechanism is not designed to replace cryptographic authentication mechanisms.

We assume that nodes in the ad hoc network may be resource constrained. Thus, in providing strong authentication for MAC layers, we use efficient symmetric cryptography, rather than relying on expensive asymmetric cryptographic operations. Especially on CPU-limited devices, symmetric cryptographic operations (such as block ciphers and hash functions) are three to four orders of magnitude faster than asymmetric cryptographic operations (such as digital signatures).

We assume that a node can obtain an authenticated key for any other node. Like public keys in systems using asymmetric cryptography, these keys in TIK are public values (once disclosed), although TIK uses only symmetric (not asymmetric) cryptography. A traditional approach to this authenticated key distribution problem is to build on a public key system for key distribution; a trusted entity can sign public-key certificates for each node, and the nodes can then use their public-key to sign new a new (symmetric) key being distributed for use in TIK. Zhou and Haas [48] propose such a public key infrastructure; Hubaux, Buttyán, and Čapkun bootstrap trust relationships from PGP-like certificates without relying on a trusted public key infrastructure [18]; Kong et al [23] propose asymmetric mechanisms for threshold signatures for certificates. Alternatively, a trusted node can securely distribute an authenticated TIK key using only symmetric-key cryptography [35] or non-cryptographic approaches [44].

We assume that all nodes in the ad hoc network have synchronized clocks with each other such that maximum bound between any two nodes' clocks is Δ. The value of the parameter Δ must be known by all nodes in the network. For the purpos of this paper, Δ must be very small; generally on the order of a few microseconds or even hundreds of nanoseconds. This level of time synchronization can be achieved now with off-the-shelf hardware based on LORAN-C [28], WWVB [29], or GPS [9, 46]; although such hardware is not currently a common part of ad hoc network nodes, it can be deployed in ad hoc networks today and is expected to become more widely utilized in future systems at reduced expense, size, weight, and power consumption. In addition, the time synchronization signal itself in such systems may be subject to certain attacks [13]. Esoteric hardware such as cesium-beam clocks, rubidium clocks, and hydrogen maser clocks could also be used in special applications today to provide sufficiently accurate time synchronization for months.

4 Summary of the TESLA Broadcast Authentication Protocol

We build our authentication mechanism on TESLA [34, 33], which provides secure authentication of broadcast messages. The main advantages of TESLA are that it adds only one message authentication code to each authenticated message, that it is robust to packet loss, and that it uses only symmetric cryptography, so its computation and communication overheads are low.

TESLA, like S/KEY [16, 24], uses one-way chains. A one-way chain is generated by choosing a random, secret value K_N, and by repeatedly computing a one-way hash function H, such as MD5 [41] or SHA-1 [30], to obtain $K_{N-1} = H[K_N]$, ..., $K_1 = H^{N-1}[K_N]$, $K_0 = H^N[K_N]$; the notation $H^i[K_N] = H[H[\ldots H[S]\ldots]]$, where the hash function H is applied i times. Typically, the elements from the one-way chain are used in reverse order; that is, K_0 is typically used first, K_1 next, until the last element K_N is used. The equality $K_j = H^{i-j}[K_i]$ holds whenever K_i and K_j are authentic chain elements and $j < i$. In TESLA, these one-way chain elements are used as keys to compute a message authentication code, then are disclosed to allow other nodes to check the authenticity of that message authentication code; this equality allows keys that are used later to be authenticated using keys that have been disclosed earlier, and also allows a node hearing one key to calculate all keys previously disclosed, in case one or more key disclosures were skipped or lost. Coppersmith and Jakobsson propose a storage efficient mechanism for one-way chains [10]: a one-way chain with N elements only requires $O(\log(N))$ storage and $O(\log(N))$ computation to access an element.

To maintain the security of TESLA, each node needs to know when each key of some other node is scheduled to be disclosed. We achieve this by disclosing keys at regular intervals. TESLA also requires that the sender and receivers be time synchronized, with a maximum time synchronization error of Δ. To account for propagation time through the

network, each TESLA node estimates, possibly adaptively, the maximum end-to-end transmission time between any two nodes τ. If a sender discloses a key at time t_i, and a TESLA time interval is of length I, then the sender will start using the key at time $t_i - (I + \tau + 2\Delta)$, and will disclose the key at time $t_i - (\tau + 2\Delta)$.

To authenticate a message in TESLA, the sender adds to the message a message authentication code based on a key from its one-way chain that has not yet been disclosed. The sender changes its key at regular intervals and discloses each key with some time delay; the length of time between key disclosures is called a *TESLA time interval*. When a node receives a packet authenticated with TESLA, the receiver checks if the key used to authenticate that packet is scheduled to have been disclosed yet. If so, then the receiver discards the packet, since an attacker could have used that key to alter the packet contents and recompute the authentication on the packet; otherwise (the key has not been disclosed yet), the receiver buffers the packet and waits for that key to be disclosed to verify the message authentication code.

We use a periodic interval for TESLA to allow nodes to know when keys will be disclosed. If a periodic interval were not used, some mechanism for reliably distributing updated key schedules would be needed. If this mechanism failed to deliver an update to some node extending the lifetime of some key, that node might fail to authenticate a valid frame; if it failed to deliver an update reducing the lifetime of a key, an attacker could reuse that key and forge a packet. Each such update would also need to be authenticated, and would be pure overhead. Furthermore, reliable delivery in an ad hoc network may be impossible, especially when nodes are partitioned. As a result, any change in key disclosure schedule would be equivalent to clock drift in nodes that did not hear the updated schedule. A mechanism that did not require synchronized clocks would be equivalent to a digital signature [6] (i.e., could be used to construct a digital signature), and hence would either be prohibitively computationally expensive, or would represent a significant advancement in the area of efficient digital signatures.

5 The Slot Authenticated MAC and a "Straw Man" Version of TIK

The Slot Authenticated MAC exploits the frame structure of a TDMA MAC to represent the timestamp efficiently. It relies on TESLA for broadcast authentication, which results in significantly increased latency.

In the Slot Authenticated MAC protocol, each TESLA time interval is divided into a number of slots. A node wishing to transmit must begin its transmission at the beginning of a slot, as in slotted ALOHA [1]. When a node wishes to send a packet in slot s, it includes the pair $\langle s, \text{HMAC}_{K_{ti}}(\text{"slot"} \mid\mid s)\rangle$ in the packet, where K_{ti} is the sender's TESLA key for the current time interval. The packet is accepted if it is received in the slot that it is being sent in, or before that slot (in case sender's clock is faster and the actual propagation delay is less than the clock skew), and rejected if it is not.

Based on the transmit power, radio receiver sensitivity, and environmental conditions, a wireless network transmitter has some maximum range r. Given the speed of light c, $\tau = \frac{r}{c}$ represents the worst-case propagation time for a legitimate transmission. To allow any pair of nodes within range to communicate, we choose a slot time of $\tau + \Delta$, since the receiver's clock could be as much as Δ faster than the sender's. Using this slot time definition, in some cases, however, an attacker may be able to successfully tunnel a packet as far as $2c\Delta$ beyond distance d, since the sender's clock may be Δ behind the receiver's clock, allowing $\tau + 2\Delta$ travel time. A more conservative setting of slot time length would be $\tau - \Delta$, since the receiver's clock could be as much as Δ slower than the sender's. This would ensure that any transmission over distance at most $r - 2c\Delta$ would be successfully authenticated. Unfortunately, some transmissions that travel distances between $r - 2c\Delta$ and r may be rejected.

The Slot Authenticated MAC suffers from three main problems: it requires the underlying MAC layer to be based on TDMA, it adds latency before a key is disclosed and hence before the packet is authenticated, and it requires an additional packet to be sent.

We propose here a "straw man" version of TIK, which we then improve in Section 7 to create a practical version of the protocol. Straw man TIK solves these three problems by introducing a general framework for authenticating packets based on transmission and arrival time, and by overlapping the authentication delay with the transmission of the packet to allow instant authentication. This straw man version of TIK requires somewhat more accurate time

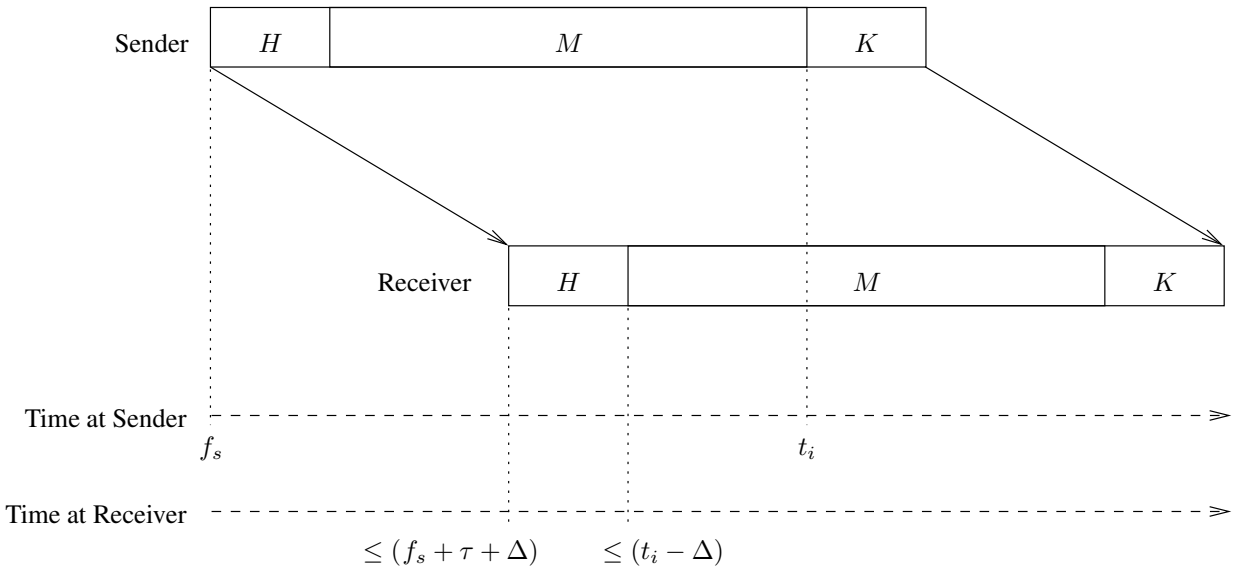

Figure 1: Timing of a frame in transmission using TIK

synchronization than TESLA, but since such accurate time synchronization is needed to detect packets that have traveled through a wormhole, we can also use the accurate time synchronization for authentication.

Figure 1 shows the timing of a frame as it is sent by the sender and received by the receiver. The time f_s here is the time at which the sender begins transmission of the frame, and time t_i is the disclosure time for some TESLA key from the sender. A straw man TIK MAC frame contains three parts: a message authentication code (shown as H in Figure 1), a message payload (shown as M), and the key used to generate the message authentication code (shown as K). A straw man TIK frame is transmitted as:

$$S \rightarrow R: \quad \langle \text{HMAC}_{K_{A_{ti}}}(M), M, K_{A_{ti}} \rangle$$

where the destination R may be unicast or broadcast.

To comply with TESLA's timing restrictions, the message authentication code must arrive at each recipient before the recipient's clock reaches time $t_i - \Delta$ (the value Δ is the maximum clock synchronization error between nodes), where t_i is the time at which the key is scheduled to be released, so that the recipient knows the sender has not yet released its key. To ensure receipt by time $t_i - \Delta$ at the recipient, allowing for the possibility that the recipient may have a clock Δ slower than the sender, and allowing for propagation delay $\tau = \frac{r}{c}$, the packet must be sent by time $t_i - \tau - 2\Delta$, where c is the speed of light in a vacuum and r is the range of the transmitter.

When this straw man version of TIK is implemented with a carrier-sensing MAC protocol (e.g., IEEE 802.11 [19]), it may not be possible to control when a MAC frame is sent relative to TESLA time intervals. In Figure 1, the key is released exactly at time t_i, but if a node cannot choose a sending time precisely, the best time interval could occur up to one time interval (I) before the key is released. In this case, the data must begin $I + \tau + 2\Delta$ before the key disclosure, which results in somewhat lower efficiency. To ensure this, given a fixed transmission rate, we choose a time interval and a minimum frame size such that the transmission time of a minimum frame size is at least $I + \tau + 2\Delta$. For example, suppose the physical layer is capable of a peak data rate of 11Mbps at a range of 300 meters, the minimum frame size is chosen to be the minimum size of an IP header (20 bytes), and the maximum time synchronization error is $1\,\mu s$. We must choose I such that every data packet will begin at least $I + \tau + 2\Delta$ before the key is disclosed; in other words, such that $I + \tau + 2\Delta \leq 14.5\,\mu s$. Since $\tau = 1\,\mu s$ and $2\Delta = 2\,\mu s$, the TESLA time interval must be chosen to be at most $11.5\,\mu s$. The amortized cost of authenticating all packets from any single node in this case is around 86,000 hashes per second. In a network of 117 nodes, over 10 million hashes per second would be needed for straw man TIK to authenticate each packet. Furthermore, though increasing the minimum frame size would reduce the computational overhead, this also reduces network efficiency and increases jitter. As a result, this straw man version

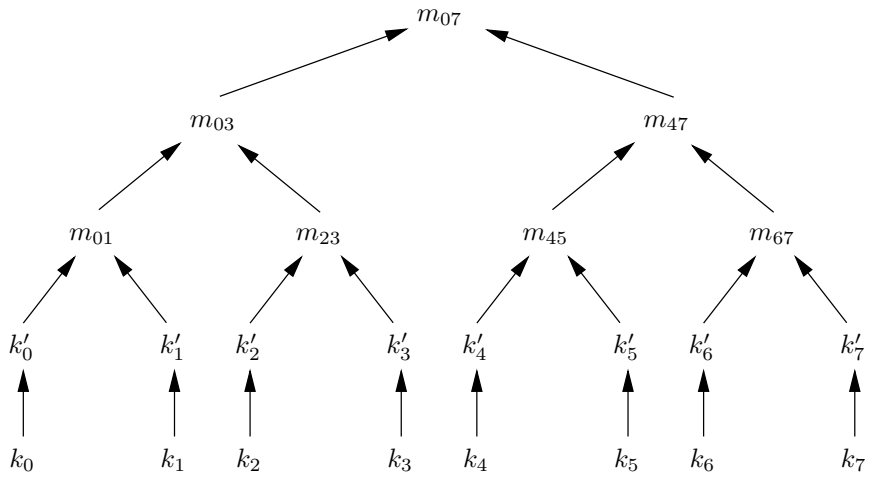

Figure 2: Merkle hash tree

of the TIK protocol is not practical in large networks. In Sections 6 and 7, we design a more efficient authentication algorithm and apply it to create a practical version of TIK.

6 Tree-Authenticated Values

Authenticating a sequence of values efficiently is an important primitive used in many security protocols. One-way hash chains are predominantly used for this purpose. One of the first usages of one-way hash chains was for one-time passwords by Lamport [24], which Haller later used to design the S/KEY one-time password system [16].

Unfortunately, as Section 5 shows, verifying values in a one-way hash chain can be computationally expensive if the number of values in the chain to be advanced over is very large. The TESLA time intervals used in straw man TIK are so short that key verification would require a substantial amount of computation. Here, we propose an extension to TESLA that has slightly higher setup cost, somewhat higher communication overhead, but much lower computational overhead for verification.

Our extension is based on the following observation: TESLA discloses keys in later packets, and if some packets are lost, the receiver might miss the verification key that corresponds to unauthenticated packets it buffers; as previously mentioned in Section 4, TESLA solves this problem by using keys from a one-way key chain, so the receiver can compute prior keys by repeatedly computing a one-way function of a key. TIK inherently tolerates packet loss (i.e., the receiver can authenticate every packet it receives), because TIK always discloses the key necessary for authentication of a packet in the packet itself. This observation allows us to use another construction to authenticate keys that is more efficient for TIK, and which may be used also in other circumstances when hash chains might be very long.

At initialization, the sender generates a sequence of keys. Since TIK only uses a small subset of the keys, we use a pseudo-random function (PRF) to derive the keys [14]. The main advantage of this method is that the sender can efficiently access the keys in any order. To establish the keys, the sender randomly chooses a secret master key $\mathcal{X}$ of the PRF $\mathcal{F}$. The sender can then easily compute each key K_i as $K_i = \mathcal{F}_{\mathcal{X}}(i)$. Without the secret master key $\mathcal{X}$, it is computationally infeasible for an attacker to derive a key K_j that the sender has not yet disclosed. The PRF function $\mathcal{F}$ could be a block cipher [15] or a message authentication code such as HMAC [4].

To authenticate the keys, we use the Merkle hash tree construction [27]. Figure 2 shows the basic Merkle hash tree construction over the eight keys $k_0, k_1, k_2, \ldots, k_7$. The sender places these keys at the leaves of a binary tree of depth 4. The sender computes each interior node of the tree as follows: a node m with child nodes $m_{\textit{left}}$ and $m_{\textit{right}}$ has a value $m = H[m_{\textit{left}} \,||\, m_{\textit{right}}]$.

The root value of the tree is used to authenticate all leaves. To authenticate a key k_i, the sender discloses i, k_i, and all the nodes necessary to verify the path up to the root. For example, if a sender wants to authenticate key k_2 in

Figure 2, it includes the values k_3, m_{01}, m_{47} in the packet. A receiver with an authentic root value m_{07} can then verify that

$$H\left[H\left[m_{01} \mid\mid H\left[k_2 \mid\mid k_3\right]\right] \mid\mid m_{47}\right]$$

equals the stored m_{07}. If the verification is successful, the receiver knows that k_2 is authentic.

The extra $k'_0, k'_1, k'_2, \ldots, k'_7$ in Figure 2 are added to the tree to avoid disclosing (in this example) the key k_3 to authenticate k_2. That is, the sender first computes a one-way function F on all keys k_i before construction the hash tree, with each $k'_i = F(k_i)$. The tree is then constructed based on the k'_i values rather than the original k_i values.

In TIK, the depth of the tree can be quite large: given a fixed time interval, a tree is of size $O(t/I)$, where t is the amount of time between rekeying. For example, if the time interval is $11.5\,\mu s$, and nodes can be rekeyed once per day, the tree is of depth 34. As a result, storing the entire tree is impractical. It is possible, however, to store only the upper layers of the tree and recompute the lower layers on demand. To reconstruct a tree of depth d requires 2^{d-1} applications of the PRF and $2^d - 1$ applications of the hash function, but saves a factor of 2^{d-1} in storage. This technique can be further improved by amortizing this calculation: a node keeps two trees of depth d: one that is fully computed and currently being used, and one that is being filled in. Since a total of $2^{d-1}+2^d-1$ operations are required to fill the tree, and the full tree will be used for 2^{d-1} time intervals, the node needs to perform only 3 operations per time interval, independent of the size of the tree.

We can now compute the true calculation and storage cost for the Merkle hash tree that we use in TIK. Let D be the depth of the entire tree, and let d be the depth of the part that is recomputed on demand. The initial computation of the tree requires 2^{D-1} evaluations of the PRF, and $2^D - 1$ evaluations of the hash function. This initial computation can be done offline, and is not time critical. To choose d, we consider the value that minimizes total storage. Since total storage is given by $2^{D-d+1} - 1 + 2 \cdot (2^d - 1)$, storage is minimized when

$$\begin{aligned}
\frac{\partial}{\partial d}\left(2^{D-d+1} - 1 + 2^{d+1} - 2\right) &= 0 \\
(-\ln 2)2^{D-d+1} + (\ln 2)2^{d+1} &= 0 \\
2^{d+1} &= 2^{D-d+1} \\
d+1 &= D-d+1
\end{aligned}$$

The optimal choice for d is $\frac{D}{2}$, and the total storage requirement is $2^{\lceil \frac{D}{2} \rceil + 1} + 2^{\lfloor \frac{D}{2} \rfloor + 1} - 3$. This represents a storage requirement of just $O(\sqrt{t/I})$. For example, a tree of depth 34 requires only 2.5 megabytes to store, much smaller than the full tree size of 170 gigabytes; once the tree is generated, it can be used at a cost of 3 operations per time interval.

A similar approach can be taken for the generation of future Merkle trees: once a single Merkle tree is generated, future Merkle trees can be generated while the original one is used for a cost of 3 hash functions per time interval plus space of $2^{\lceil \frac{D}{2} \rceil + 1} + 2^{\lfloor \frac{D}{2} \rfloor + 1} - 2$. Only the root of each new tree needs to be distributed, and as mentioned in Section 3, these values can be distributed using only symmetric-key cryptography [35] non-cryptographic approaches [44], or by sending them using the current Merkle tree for authentication.

7 TIK

When tree-authenticated values are used, the associated values of the tree must be included in each packet. In this improved version of TIK, a MAC layer frame has four fields: a message authentication code, a payload, the associated values of the tree, and the key used to generate the message authentication code. An improved TIK frame is transmitted as:

$$A \rightarrow B: \quad \langle \text{HMAC}_{K_{A_{ti}}}(M), M, \textit{tree elements}, K_{A_{ti}} \rangle$$

The Merkle hash tree provides the distinct advantages of faster and deterministic-time verification, and though it adds some network overhead, that network overhead somewhat offsets the minimum frame length.

A TDMA MAC may be able to choose the frame start time so that the message authentication code is sent by time $t_i - \frac{r}{c} - 2\Delta$. In this case, the minimum payload length is $\frac{r}{c} + 2\Delta$ times the bit rate. For additional efficiency, the

nodes should have different key disclosure times, and the MAC layer should provide each node with the MAC level time slot it needs for authenticated delivery.

As mentioned in Section 5, CSMA MACs may not be able to control when a MAC frame is sent relative to TESLA time intervals. In this case, the minimum frame size needs to be chosen so that a time interval boundary is guaranteed to exist somewhere inside the packet. For example, if the physical layer is capable of a peak data rate of 100Mbps and a range of 150 meters, the TESLA time interval is chosen to be $25\ \mu s$, and time synchronization is achieved to within $250\ ns$, then the minimum frame size must be at least 325 bytes. However, if each element in the Merkle hash tree is 80 bits long, and the depth of the tree is 31, then the minimum frame size is just 15 bytes.

If a MAC protocol uses an Request-to-Send/Clear-to-Send (RTS/CTS) handshake, the minimum frame size can be reduced by carrying the message authentication code inside the RTS frame:

$$\begin{aligned} A \rightarrow B: &\quad \langle \textit{RTS}, \text{HMAC}_{K_{A_{ti}}}(M) \rangle \\ B \rightarrow A: &\quad \langle \textit{CTS} \rangle \\ A \rightarrow B: &\quad \langle \textit{DATA}, M, \textit{tree elements}, K_{A_{ti}} \rangle \end{aligned}$$

In particular, instead of having a minimum payload length of $\frac{r}{c} + 2\Delta + I$ times the data rate, where I is the duration of a time interval, the minimum payload length is just $2\Delta + I - 2t_{turn}$ times the data rate, where t_{turn} is the minimum allowed time between receiving a control frame and returning a corresponding frame. This minimum payload length includes the length of the CTS, DATA header, data, and tree elements.

To use TIK to prevent wormholes, we add a timestamp to each packet, identifying the time at which the sender first started transmitting this packet. For example, each packet could include a 64-bit timestamp with nanosecond resolution, allowing over 580 years of use starting from the epoch. Since the entire packet is authenticated, the timestamp is authenticated. The recipient can compare the timestamp to the time at which the packet was received and determine the transmission delay, plus or minus the clock synchronization error, and decide whether to accept or reject the packet. As in the Slot Authenticated MAC protocol (Section 5), a policy could be set allowing the reception of packets for which the perceived transmission delay, or arrival time minus sending timestamp, is less than some threshold. That threshold could be chosen anywhere between $\tau - \Delta$ and $\tau + \Delta$, where the more conservative approach of $\tau - \Delta$ never allows tunnels but rejects some valid packets, and the more liberal approach of $\tau + \Delta$ never rejects valid packets, but may allow tunneling of up to $2c\Delta$ past the actual transmission range.

With a GPS-disciplined clock [46], time synchronization to within $\Delta = 183ns$ with probability $1-10^{-10}$. If a transmitter has a 250 meter range, the $\tau - \Delta$ threshold accepts all packets sent less than 140 meters and some packets sent between 140 and 250 meters; the $\tau+\Delta$ threshold accepts all packets sent less than 250 meters but allows tunneling of packets up to 110 meters beyond that.

8 Discussion

To evaluate the suitability of a protocol for use in ad hoc networks, we measured computational power and memory currently available in mobile devices. To measure the number of repeated hashes that can be performed per second, we optimized the MD5 code from ISI [45] to achieve maximum performance for repeated hashing. Our optimized version performs 10 million hash function evaluations in 7.544 seconds on a Pentium III running at 1 GHz, representing a rate of 1.3 million hashes per second; the same number of hashs using this implementation on a Compaq iPaq 3870 PocketPC running Linux took 45 seconds, representing a rate of 222,000 hashes per second. Repetitive, simple functions like hashes can also be efficiently implemented in hardware; Helion Technology [17] claims a 20k gate ASIC core design (a third the complexity of Bluetooth [3] and less than a third the complexity of IEEE 802.11 [22]) capable of more than 1.9 million hashes per second and a Xilinx FPGA design using 1650 LUTs capable of 1 million hashes per second. In terms of memory consumption, existing handheld devices, such as the iPaq 3870, come equipped with 32 MB of Flash and 64 MB of RAM. Modern notebooks can generally be equipped with hundreds of megabytes of RAM.

A high-end wireless LAN such as the Proxim Harmony 802.11a [38] has range potentially as far as 250 meters and data rate as high as 108 Mbps. With time synchronization provided by a Trimble Thunderbolt GPS-Disciplined

Clock [46], the synchronization error can be as low as 183 ns with probability $1-10^{-10}$. If authentic keys are re-established every day, with a 20 byte minimum packet size and an 80-bit message authentication code length, the tree has depth 33, giving a minimum frame length of 350 bytes, or $25.9\,\mu s$, and a time interval of $24.7\,\mu s$. Assuming that the node generates new trees for redistribution while it is using its old trees, it requires 8 megabytes of storage and needs to perform fewer than 243,000 operations per second to maintain and generate trees. To authenticate a received packet, a node needs perform only 33 hash functions. To keep up with link-speed, a node needs to verify a packet every $25.9\,\mu s$, thus requiring 1,273,000 hashes per second, for a total computational requirement of 1,516,000 hashes per second. This can be achieved today in hardware, either by placing two MD5 units on a single FPGA, or with an ASIC. High-end laptops today sport 1.2 GHz Pentium III CPUs, which should also be able to perform 1.5 million hash operations per second.

Current commodity wireless LAN products such as commonly used 802.11b cards [2] provide 11 Mbps at 250 meters. Given the same time synchronization, rekeying interval, minimum packet size, and message authentication code length, the tree has depth 30, giving a minimum frame length of 320 bytes, or $232\,\mu s$, and a time interval of $231.5\,\mu s$. Assuming that the node generates new trees for redistribution while it is using its old trees, it requires just 2.6 megabytes of storage and needs to perform just 26,500 operations per second. To authenticate a received packet, a node needs perform only 30 hash functions. Since any IP packet authenticated using TIK would take at least $232\,\mu s$ to transmit, TIK can authenticate packets at link-speed using just 13,000 hashes per second, for a total of 39,500 hash functions per second, which is well within the capability of an iPaq, with 82.2% of its CPU time to spare.

In a sensor network such as Hollar et al's weC mote [21, 47], nodes may only be able to achieve time synchronization accurate to 1 second, have a 19.6 kbps link speed, and 20 meter range. In this case, the smallest packet that can be authenticated is 4900 bytes; since the weC mote does not have sufficient memory to store this packet, TIK is unusable in such a resource-scarce system. Furthermore, the level of time synchronization in this system is such that neither the Slot Authenticated MAC nor TIK could provide a usable wormhole detection system.

9 Leashes: A Generalized Approach to Wormhole Detection

In this section, we discuss general mechanisms to detect wormholes, based on traditional authentication and signature systems.

To generalize the mechanisms for wormhole detection, we introduce the notion of a *leash*. A leash is any information that is added to a packet designed to restrict the maximum allowed transmission distance. A leash can prevent the wormhole attack, because it allows the receiver of a packet to detect if the packet traveled further than the leash allows. We give two examples of how to construct a leash, using a timestamp or location information in the packet, in conjunction with either an authentication or a signature scheme.

A leash using a timestamp requires accurate time synchronization, so that the receiver can detect if the packet traveled past the distance restricted by the leash; this approach is similar to TIK. If the sender places a timestamp in the packet when it is sent, a receiver is able to detect if the packet traveled too far, based on the claimed transmission time and the speed of light. A regular digital signature scheme, e.g., RSA [40], or other authentication technique, can be used to allow a receiver to authenticate the timestamp.

One possible problem with using a timestamp in the packet is that in a contention-based MAC, the sender may not know the precise time at which it will send a packet. For example, a sender using the IEEE 802.11 MAC may not know the time a packet will be transmitted until approximately one slot time (20 μs) prior to transmission. Generating an inefficient digital signature, such as RSA with a 1024 bit key, could take three orders of magnitude more time than this slot time (on the order of 10 ms). The sender can use two approaches to hide the signature generation latency: either increase the MTU (minimum transmission unit) to allow computation to overlap with transmission, or use a more efficient signature scheme, such as Schnorr's signature [42], which enables efficient signature generation after pre-processing.

Another method to construct a leash is to use location information and loosely synchronized clocks. If the clocks of the sender and receiver are synchronized to within $\pm\Delta$, and ν is an upper bound on the velocity of any node, then the receiver can compute an upper bound on the distance between the sender and itself d_{sr}. Specifically, based on the

timestamp t_s in the packet, the local receive time t_r, the maximum relative error in location information δ, and the locations of the receiver p_r and the sender p_s, d_{sr} can be bounded by $d_{sr} \leq ||p_s - p_r|| + 2\nu \cdot (t_r - t_s + \Delta) + \delta$.

In certain circumstances, bounding the distance between the sender and receiver d_{sr} cannot prevent wormhole attacks; for example, when obstacles prevent communication between two nodes that would otherwise be in transmission range, a distance-based scheme would still allow wormholes between the sender and receiver. A network that uses location information as a leash can control even these kinds of wormholes. To accomplish this, each node has a radio propogation model. A receiver verifies that every possible location of the sender (a $\delta + \nu(t_r - t_s + 2\Delta)$ radius around p_s) can reach every possible location of the receiver (a $\delta + \nu(t_r - t_s + 2\Delta)$ radius around p_r).

An advantage to using the location information approach to constructing a leash, in conjunction with a signature scheme (i.e., a signature providing non-repudiation), is that an attacker can be caught if it pretends to reside at multiple locations. This use of non-repudiation was also proposed by Sirois and Kent [43]. When a legitimate node overhears the attacker claiming to be in different locations that would only be possible if the attacker could travel at a velocity above the maximum node velocity ν, the legitimate node can use the signed locations to convince other legitimate nodes that the attacker is malicious. In particular, if some node claims to be at locations p_1 and p_2 at times t_1 and t_2, respectively, and if $\frac{||p_2 - p_1||}{|t_2 - t_1|} > \nu$, a legitimate node can broadcast these two packets to convince other nodes that the first node is indeed an attacker. Each node hearing these messages can check the two signatures, verify the discrepancy in the information, and rebroadcast the information if it has not previously done so. To easily perform duplicate suppression in rebroadcasting this information, each node can maintain a *blacklist*. Each blacklist entry contains a node address and the time the blacklist entry expires. When a node receives a message showing an attacker's behavior, it checks if that attacker is already on its blacklist; if so, it updates the expiration time on its current entry and discards the new message. Otherwise, it adds a new blacklist entry and propagates the message.

10 Related Work

Radio Frequency watermarking is another possible approach to providing the security described in this paper. Since we are aware of no published specific details, it is difficult to assess its security. If the radio hardware is kept secret, such as through tamper-resistant modules, some level of security can be provided against compromised nodes; however, if the radio band in which communications are taking place is known, then an attacker can attempt to tunnel the entire signal from one location to another.

It may be possible to modify existing intrusion detection approaches to detect a wormhole attacker; since the packets sent by the wormhole are identical to the packets sent by legitimate nodes, such detection would more easily be achieved jointly with hardware able to specify some sort of directionality information for received packets. To the best of our knowledge, no work has been published regarding the possibility of using intrusion detection systems specifically to detect wormhole attackers.

TESLA generally chooses longer time intervals than TIK to reduce the amount of computation needed to authenticate a new key. As a result, TESLA is capable of functioning with much looser time synchronization than TIK requires. Given a sufficient level of time synchronization, TIK provides an advantage over hop-by-hop authentication with TESLA, with respect to latency and packet overhead, but it suffers with respect to byte overhead. In particular, since TIK key disclosure always occurs in the same packet as the data protected, packets can be verified instantly, whereas with TESLA, packets must wait, on average 1.5 time intervals, which is especially significant when packets are authenticated hop-by-hop in a multi-hop ad hoc network routing protocol.

The IEEE 802.11i Task Group is designing modifications to IEEE 802.11 [19] to improve security. These modifications generally use a single shared key, or, when multiple keys are used, the keys are used between multiple clients and a single base station. Since base stations are not present in ad hoc networks, and since a single shared key does not prevent any attacks launched from a compromised node, these proposals do not sufficiently address authentication for ad hoc network routing. Furthermore, none of the current proposals within IEEE 802.11i address the wormhole attack.

Other Medium Access Control protocols specify privacy and authenticity mechanisms. These mechanisms typically use one or more shared keys, allowing compromised nodes to forge packets. Furthermore, to the best of our knowledge, none of these mechanisms protect against wormhole attacks.

11 Conclusions

In this paper, we described the wormhole attack and how many proposed ad hoc network routing protocols are vulnerable to this attack. We have presented TIK, a security layer on top of existing MAC layers, which provides instant authentication of received packets. TIK requires just $O(n)$ keys, $O(\sqrt{t/I})$ storage cost, $O(\log t/I)$ verification cost, $O(1)$ computation per time interval, and $O(\log t/I)$ per-packet overhead, where n is the number of nodes in the network and t is the amount of time between key redistributions. In particular, a node need only perform between 3 and 6 hash function evaluations per time interval to maintain up-to-date key information for itself, and roughly 30 hash functions for each received packet. With commodity technology such as 11 Mbps wireless links, TIK has computational and memory requirements that are easily satisfiable today; 2.6 megabytes for tree storage represents, for example, less than 3% of the standard memory on an Compaq iPaq 3870 with no external memory cards, and since the StrongARM on the iPaq is capable of performing 222,000 symmetric cryptographic operations per second, TIK imposes no more than a 18% load on the CPU, even when flooded with packets, and often less than that in normal operation.

When used in conjunction with precise timestamps and very accurate time synchronization, TIK can prevent wormhole attacks that cause the signal to travel a distance longer than the nominal range of the radio, or any other range that might be specified. Sufficiently tight time synchronization can be achieved in a wireless LAN using commercial GPS receivers [46], and wireless MAN technology could be sufficiently time-synchronized using either GPS or LORAN-C [28] radio signals.

A MAC using TIK efficiently protects against replay, spoofing, and wormhole attacks, and ensures strong freshness. TIK is implementable with current technologies, and does not require significant additional processing at the MAC layer, since the authentication of each packet can be performed at the host CPU.

Finally, our approach can be generalized for use with arbitrary authentication and signature schemes. We introduce the concept of a *leash*, which is an authenticated piece of data that limits transmission range between two legitimate hosts. We present a scheme that allows detection of certain attacks and blacklisting of attackers when signatures are used.

References

[1] Norman Abramson. The ALOHA System—Another Alternative for Computer Communications. In *Proceedings of the Fall 1970 AFIPS Computer Conference*, pages 281–285, November 1970.

[2] Agere Systems Inc. Specifications for ORiNOCO World PC Card. Allentown, PA. Available at `ftp://ftp.orinocowireless.com/pub/docs/ORINOCO/BROCHURES/US/WorldPCCardUS.pdf`.

[3] ARC International. ARC releases BlueForm, a comprehensive solution for Bluetooth systems on a chip. Elstree, United Kingdom. Available at `http://www.arccores.com/newsevents/PR/6-04-01-2.htm`, June 2001.

[4] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying Hash Functions for Message Authentication. In *Advances in Cryptology – CRYPTO '96*, edited by Neal Koblitz, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, Berlin Germany, 1996.

[5] Bhargav Bellur and Richard G. Ogier. A reliable, efficient topology broadcast protocol for dynamic networks. In *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 178–186, March 1999.

[6] Dan Boneh, Glenn Durfee, and Matt Franklin. Lower Bounds for Multicast Message Authentication. In *Advances in Cryptology – EUROCRYPT '2001*, edited by Birgit Pfitzmann, volume 2045 of *Lecture Notes in Computer Science*, pages 434–450, Innsbruck, Austria, 2001. Springer-Verlag, Berlin Germany.

[7] Michael Brown, Donny Cheung, Darrel Hankerson, Julio Lopez Hernandez, Michael Kirkup, and Alfred Menezes. PGP in Constrained Wireless Devices. In *Proceedings of the 9th USENIX Security Symposium*, Denver, Colorado, August 2000.

[8] Ran Canetti, Juan Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas. Multicast Security: A Taxonomy and Some Efficient Constructions. In *Proceedings of INFOCOMM'99*, March 1999.

[9] Tom Clark. Tom Clark's Totally Accurate Clock FTP Site. Greenbelt, Maryland. Available at `ftp://aleph.gsfc.nasa.gov/GPS/totally.accurate.clock/`.

[10] Don Coppersmith and Markus Jakobsson. Almost Optimal Hash Sequence Traversal. In *Proceedings of the Fifth Conference on Financial Cryptography (FC '02)*, February 2002.

[11] Nicolas Courtois, Louis Goubin, and Jacques Patarin. Flash, a fast multivariate signature algorithm. In *Progress in Cryptology - CT-RSA 2001*, edited by David Naccache, volume 2020 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, April 2001.

[12] Defense Advanced Research Projects Agency. Frequently Asked Questions v4 for BAA 01-01, FCS Communications Technology. Washington, DC. Available at `http://www.darpa.mil/ato/solicit/baa01_01faqv4.doc`, October 2000.

[13] Eran Gabber and Avishai Wool. How to Prove Where You Are. In *Proceedings of the 5th ACM Conference on Computer and communications Security*, pages 142–149, November 1998.

[14] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[15] Shafi Goldwasser and Mihir Bellare. Lecture Notes on Cryptography. Summer Course "Cryptography and Computer Security" at MIT, 1996–1999, August 1999.

[16] Neil M. Haller. The S/KEY One-time Password System. In *Proceedings of the Symposium on Network and Distributed Systems Security*, edited by Dan Nesset and Robj Shirey, San Diego, California, February 1994.

[17] Helion Technology Ltd. High performance Solutions in Silicon — MD5 core. Cambridge, England. Available at `http://www.heliontech.com/core5.htm`.

[18] Jean-Pierre Hubaux, Levente Buttyán, and Srdjan Čapkun. The Quest for Security in Mobile Ad Hoc Networks. In *ACM Symposium on Mobile Ad Hoc Networking and Computing* (MobiHoc 2001), Long Beach, CA, USA, October 2001.

[19] IEEE Computer Society LAN MAN Standards Committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std 802.11-1997. The Institute of Electrical and Electronics Engineers, New York, New York, 1997.

[20] David B. Johnson, David A. Maltz, and Josh Broch. The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks. In *Ad Hoc Networking*, edited by Charles E. Perkins, chapter 5, pages 139–172. Addison-Wesley, 2001.

[21] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: mobile networking for Smart Dust. In *International Conference on Mobile Computing and Networking (MobiCom '99)*, pages 271–278, August 1999.

[22] Dean Kawaguchi and Sarosh Vesuna. Symbol Technologies, Inc. Automates Ssystem-To-Gates Design Flow For Wireless LAN ASIC with COSSAP and Behavioral Compiler. Mountain View, California. Available at `http://www.synopsys.com/news/pubs/bctb/sep98/frame_art1.html`, September 1998.

[23] Jiejun Konh, Petros Zerfos, Haiyun Luo, Songwu Lu, and Lixia Zhang. Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks. In *Ninth International Conference on Network Protocols (ICNP '01)*, pages 251–260, November 2001.

[24] Leslie Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–772, November 1981.

[25] Arjen K. Lenstra and Eric R. Verheul. Selecting Cryptographic Key Sizes. Available at `http://www.cryptosavvy.com/`, November 1999. A shorter version of the report appeared in the proceedings of the Public Key Cryptography Conference (PKC2000) and in the Autumn '99 PricewaterhouseCoopers CCE newsletter. A revised version appeared later in the Journal of Cryptology.

[26] Arjen K. Lenstra and Eric R. Verheul. Key Improvements to XTR. In *Advances in Cryptology – ASIACRYPT '2000*, edited by T. Okamoto, volume 1976 of *Lecture Notes in Computer Science*, pages 220–233, Kyoto, Japan, 2000. Springer-Verlag, Berlin Germany.

[27] Ralph Merkle. Protocols for Public Key Cryptosystems. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, April 1980.

[28] David L. Mills. A Computer-Controlled LORAN-C Receiver for Precision Timekeeping. Technical Report 92-3-1, Department of Electrical and Computer Engineering, University of Delaware, March 1992.

[29] David L. Mills. A Precision Radio Clock for WWV Transmissions. Technical Report 97-8-1, Department of Electrical and Computer Engineering, University of Delaware, August 1997.

[30] National Institute of Standards and Technology (NIST). Secure Hash Standard, May 1993. Federal Information Processing Standards (FIPS) Publication 180-1.

[31] Charles E. Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, August 1994.

[32] Charles E. Perkins and Elizabeth M. Royer. Ad-Hoc On-Demand Distance Vector Routing. In *Second IEEE Workshop on Mobile Computing Systems and Applications* (WMCSA'99), pages 90–100, February 1999.

[33] Adrian Perrig, Ran Canetti, Dawn Song, and Doug Tygar. Efficient and Secure Source Authentication for Multicast. In *Network and Distributed System Security Symposium, NDSS '01*, pages 35–46, February 2001.

[34] Adrian Perrig, Ran Canetti, Doug Tygar, and Dawn Song. Efficient Authentication and Signature of Multicast Streams over Lossy Channels. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 56–73, Oakland, CA, May 2000.

[35] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. SPINS: Security Protocols for Sensor Networks. In *Seventh Annual ACM International Conference on Mobile Computing and Networks (MobiCom 2001)*, Rome, Italy, July 2001.

[36] Raymond L. Pickholtz, Donald L. Schilling, and Laurence B. Milstein. Theory of Spread Spectrum Communications — A Tutorial. *IEEE Transactions on Communications*, 30(5):855–884, May 1982.

[37] Guillaume Poupard and Jacques Stern. On The Fly Signatures based on Factoring. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, pages 37–45, Singapore, November 1999.

[38] Proxim, Inc. Data sheet for Proxim Harmony 802.11a CardBus Card. Sunnyvale, CA. Available at `http://www.proxim.com/products/all/harmony/docs/ds/harmony_11a_cardbus.%pdf`.

[39] Amir Qayyum, Laurent Viennot, and Anis Laouiti. Multipoint Relaying: An Efficient Technique for flooding in Mobile Wireless Networks. Technical Report Research Report RR-3898, INRIA, February 2000.

[40] Ron L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[41] Ronald L. Rivest. The MD5 Message-Digest Algorithm. Internet Request for Comment RFC 1321, Internet Engineering Task Force, April 1992.

[42] Claus P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[43] Karen E. Sirois and Stephen T. Kent. Securing the Nimrod Routing Architecture. In *Symposium on Network and Distributed Systems Security (NDSS '97)*, San Diego, California, February 1997.

[44] Frank Stajano and Ross Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In *Security Protocols, 7th International Workshop*, edited by B. Christianson, B. Crispo, and M. Roe. Springer-Verlag, Berlin Germany, 1999.

[45] Joseph D. Touch. Performance Analysis of MD5. In *Proceedings of the ACM SIGCOMM '95 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 77–86, August 1995.

[46] Trimble Navigation Limited. Data Sheet and Specifications for Trimble Thunderbolt GPS Disciplined Clock. Sunnyvale, California. Available at `http://www.trimble.com/thunderbolt.html`.

[47] Alec Woo. CS294-8 Deeply Networked Systems Mote Documentation and Development Information. Berkeley, CA. Available at `http://www.cs.berkeley.edu/~awoo/smartdust/`.

[48] Lidong Zhou and Zygmunt J. Haas. Securing Ad Hoc Networks. *IEEE Network Magazine*, 13(6):24–30, November/December 1999.